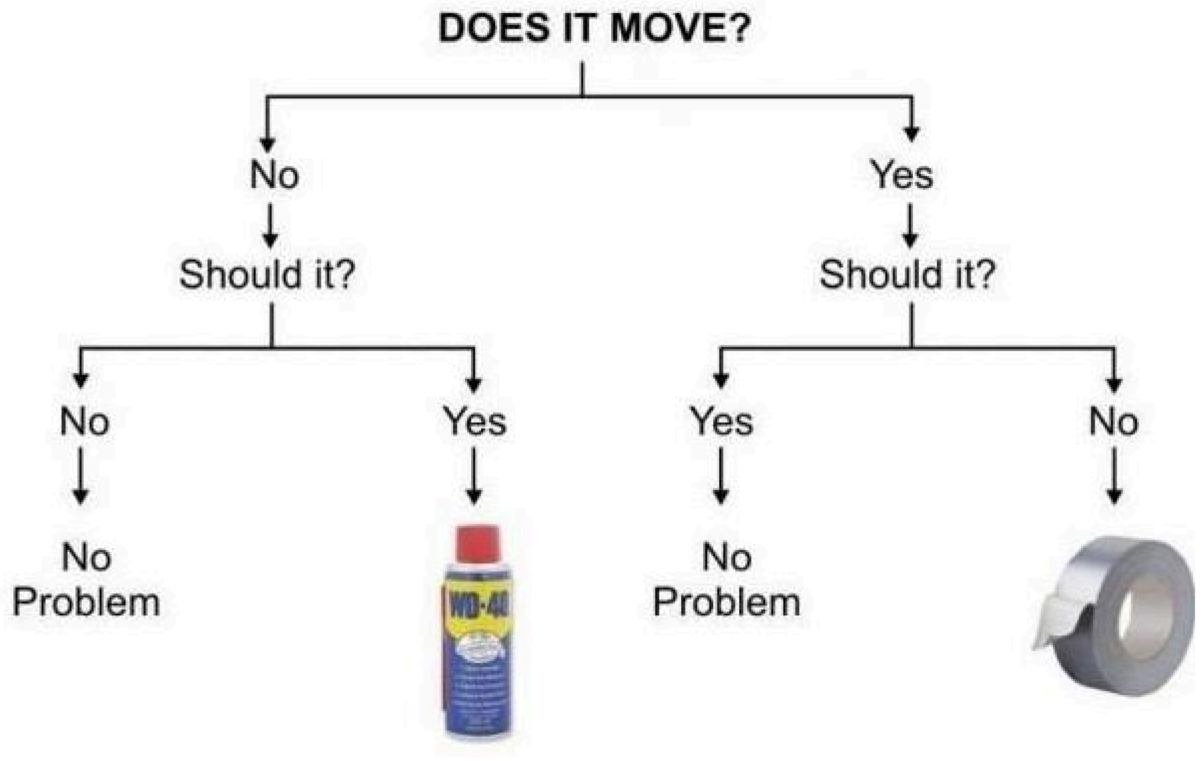
ELEC 5260/6260
Embedded Computing Systems

Spring 2012

Victor P. Nelson

Text: "Computers as Components, 2nd Edition"
Prof. Wayne Wolf (Georgia Tech)

Engineering Flowchart



Course Topics

- Embedded system design & modeling
  - The embedded computing space.
- System design methodologies
- Platforms: system-on-chip, microcontrollers, networks.
  - CPUs for embedded systems (ARM,SHARC)
  - uCdragon ARM development board
- Architectures, applications, methodologies.
  - Hardware, software, system.
- Embedded program design

(continued)

Course Topics (continued)

- Input/output devices & interrupts
- Real-time embedded operating systems
- Multiprocessor embedded networks
- Standards-based design.
- Case studies

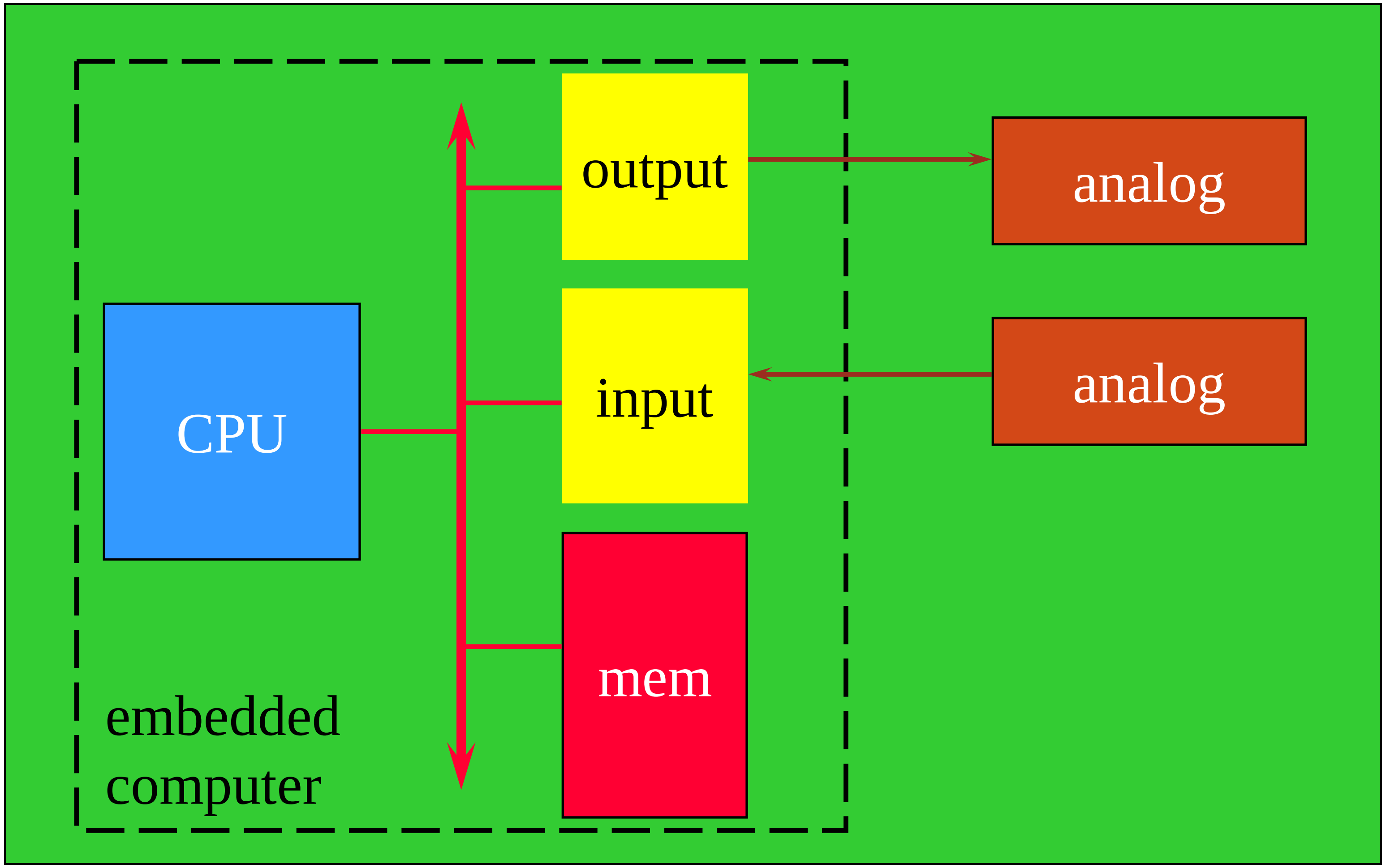
Not simply a "microcontroller course".

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:
  - dedicated to specific function(s)
  - don't need all the general-purpose bells and whistles.
  - interact with environment

Embedding a computer



Examples

- Personal digital assistant (PDA).
- Printer.
- Disk drive: motor control, data xfer, etc.
- Cell phone/“smart” phones”.
- Digital cameras – still and video
- Automobile: engine, brakes, dash, etc.
- Television, CD/DVD/MP3 player.
- Household appliances.
- PC keyboard (key scanning).
- Industrial process control.
- Avionics & aerospace vehicles

Early history

- Late 1940's: MIT Whirlwind computer was designed for real-time operations.
  - Originally designed to control an aircraft simulator.
- First microprocessor was Intel 4004 in early 1970's.
- HP-35 calculator used several chips to implement a microprocessor in 1972.
- 4-bit microcontrollers created in the 1970's
- 8-bit microcontrollers in mid 1970's

Early history, cont'd.

- 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.
- High-performance 32-bit microcontrollers enable movement of functions from HW to SW
  - Radio.
  - Multimedia.
  - Communications
  - Desktop PC applications.
  - Complex control.

Microprocessor varieties

- **Microcontroller:** includes I/O devices, on-board memory.
- **Digital signal processor (DSP):** microprocessor optimized for digital signal processing.
- **Application-Specific Processor (ASP):** instruction set & architecture tailored to application (graphics, network, etc.)
- **"Soft core"** – microcontroller or CPU to be embedded in a system on chip (SoC), including "platform FPGA"

- Typical embedded word sizes:

  4-bit, 8-bit, 16-bit, 32-bit.

Application examples

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

- ASSIGNMENT #1: 4-5 page report on a current multimedia system/device

Automotive embedded systems

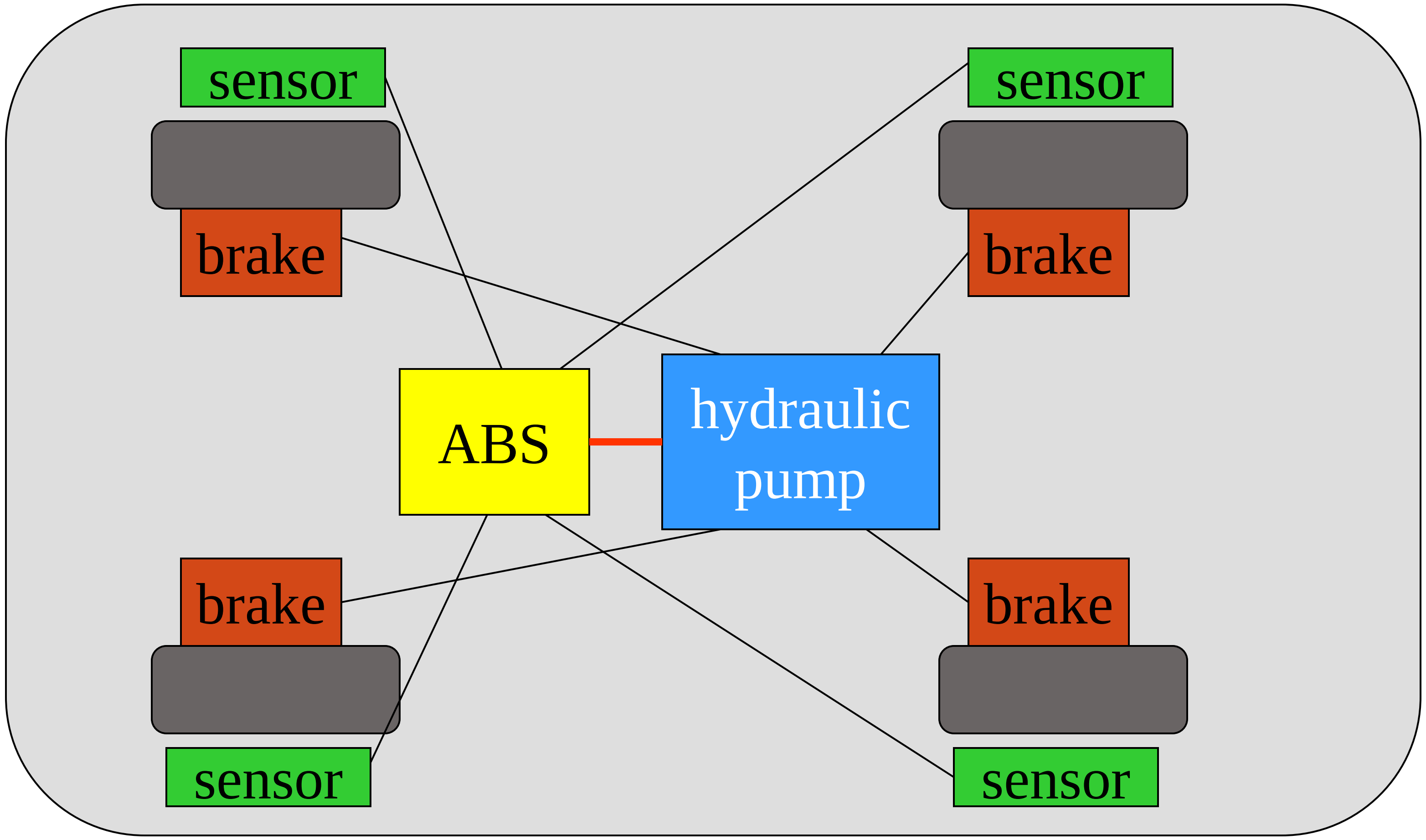
- Today's high-end automobile may have 100 microprocessors:
  - 4-bit microcontroller checks seat belt;
  - microcontrollers run dashboard devices;
  - 16/32-bit microprocessor controls engine.
  - Network of microcontrollers control antilock brakes

BMW 850i brake and stability control system

- **Anti-lock brake system (ABS):** pumps brakes to reduce skidding.
- **Automatic stability control + traction (ASC+T):** controls engine to improve stability (throttle, ignition timing, differential brake, gears).
- ABS and ASC+T communicate.
  - ABS was introduced first---needed to interface to existing ABS module.

Diagram – next slide

BMW 850i, cont'd.



Characteristics of high-end embedded systems

- Complex algorithms: high performance & functionality.
- High data rates
- Large data structures
- Varied user interfaces.
- Multiple task, heterogeneous.
- Real-time.
- Often low power.
- Manufacturing: cost-effective, reliable.

Often have to make trade-offs in characteristics

Project Cost

- Total cost of a project involves non-recurring engineering (NRE), cost plus recurring (RE) cost, and number of units produced (K)

$$\text{Project Cost} = \text{NRE} + \text{K*RE}$$

  - NRE includes design time, tools, facilities
  - RE includes components, manufacturing, testing, and maintenance

Real-time operation

- Must finish operations by deadlines.
  - **Hard real time:** missing deadline causes failure.
  - **Soft real time:** missing deadline results in degraded performance.
- Many systems are **multi-rate:** must handle operations at widely varying rates.

Power considerations

- Custom logic is a clear winner for low power devices.
- Modern microprocessors offer features to help control power consumption.
  - Turn off unnecessary logic
  - Reduce memory accesses
  - Reduce clock rates (CMOS)
- Software design techniques can help reduce power consumption.

Design development time

- Often must meet tight deadlines.
  - 6 month market window is common.
  - Can’t miss back-to-school window for calculator or Christmas holiday sales window for “toys”
  - Employ hardware-software codesign to shorten design time

Why use microprocessors?

- Microprocessors are often very efficient: can use same logic to perform many different functions.
- Microprocessors simplify upgrades and the design of families of products.
- Alternatives: custom logic implemented with field-programmable gate arrays (FPGAs), ASICs, etc.
- "Platform" FPGAs – implement one or more microprocessor cores, with embedded memory and programmable logic

The performance paradox

- Microprocessors use much more logic to implement a function than does custom logic.
- <u>But</u> microprocessors are often at least as fast:
  - aggressive VLSI technology;
  - heavily pipelined;
  - smart compilers;
  - re-use of efficient SW routines.

Execution Time =

(#instructions) x (#clocks/instruction) x (clock period)

Challenges in embedded system design

- How much hardware do we need?
  - How big is the CPU? Memory?
  - What peripheral functions?
    - HW vs SW
- How do we meet our deadlines?
  - Faster hardware or cleverer software?
  - Real-time operating system or custom design?
- How do we minimize power?
- How do we optimize cost?