Designing with microprocessors

- Architectures and components:
  - software;
  - hardware.
- Debugging.
- Manufacturing testing.

Hardware platform architecture

Contains several elements:

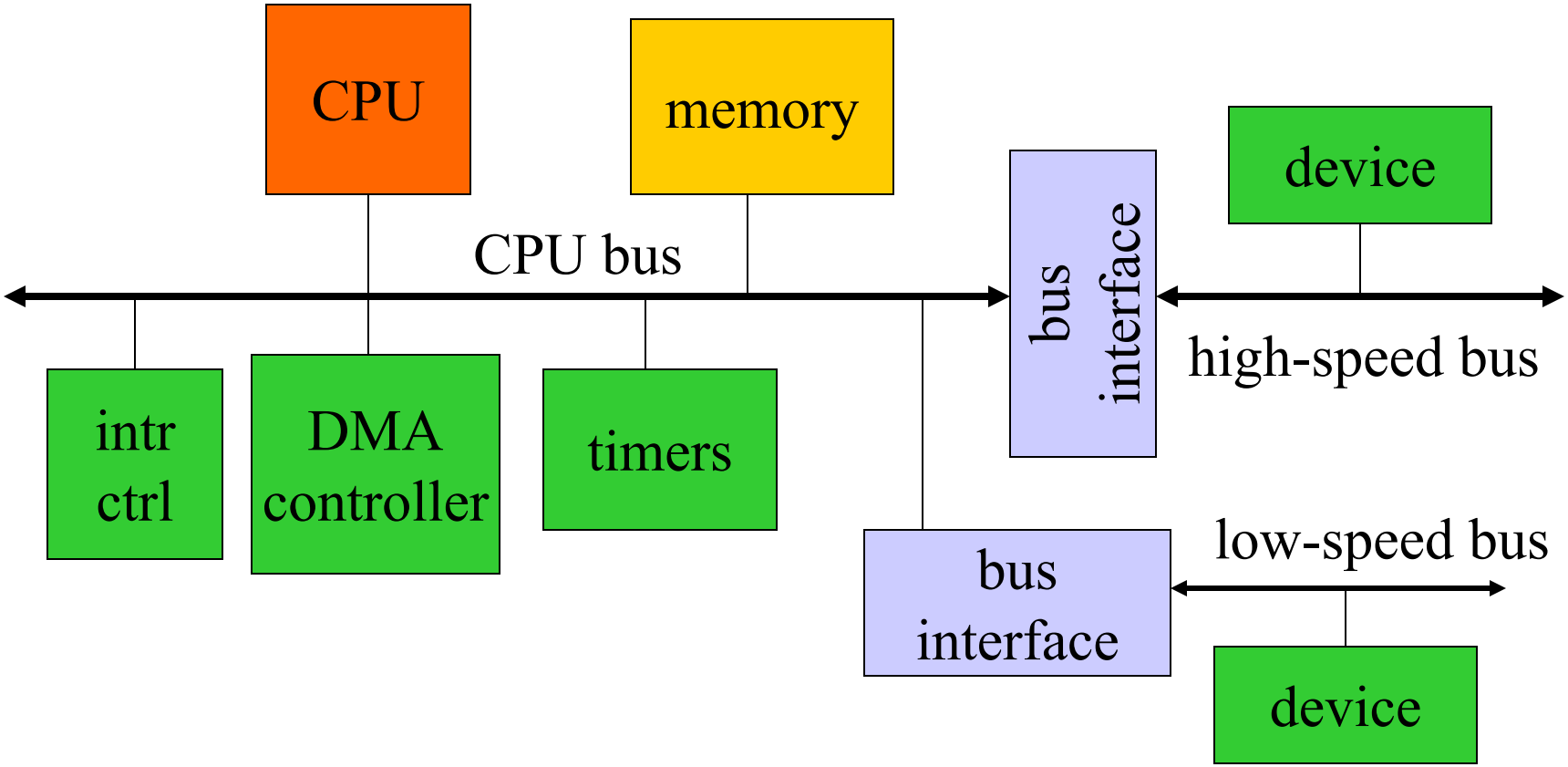
- CPU;
- bus;
- memory;
- I/O devices: networking, sensors, actuators, etc.

How big/fast much each one be?

The PC as a platform

- Advantages:
  - cheap and easy to get;
  - rich and familiar software environment.
- Disadvantages:
  - requires a lot of hardware resources;
  - not well-adapted to real-time.

Typical PC hardware platform



Typical busses

- ISA (Industry Standard Architecture): original IBM PC bus, low-speed by today's standard.
- PCI: standard for high-speed interfacing
  - 33 or 66 MHz.
- USB (Universal Serial Bus), Firewire: relatively low-cost serial interface with high speed.

PC platform software

- IBM PC uses BIOS (Basic I/O System) to implement low-level functions:
  - boot-up;
  - minimal device drivers.
- BIOS has become a generic term for the lowest-level system software.

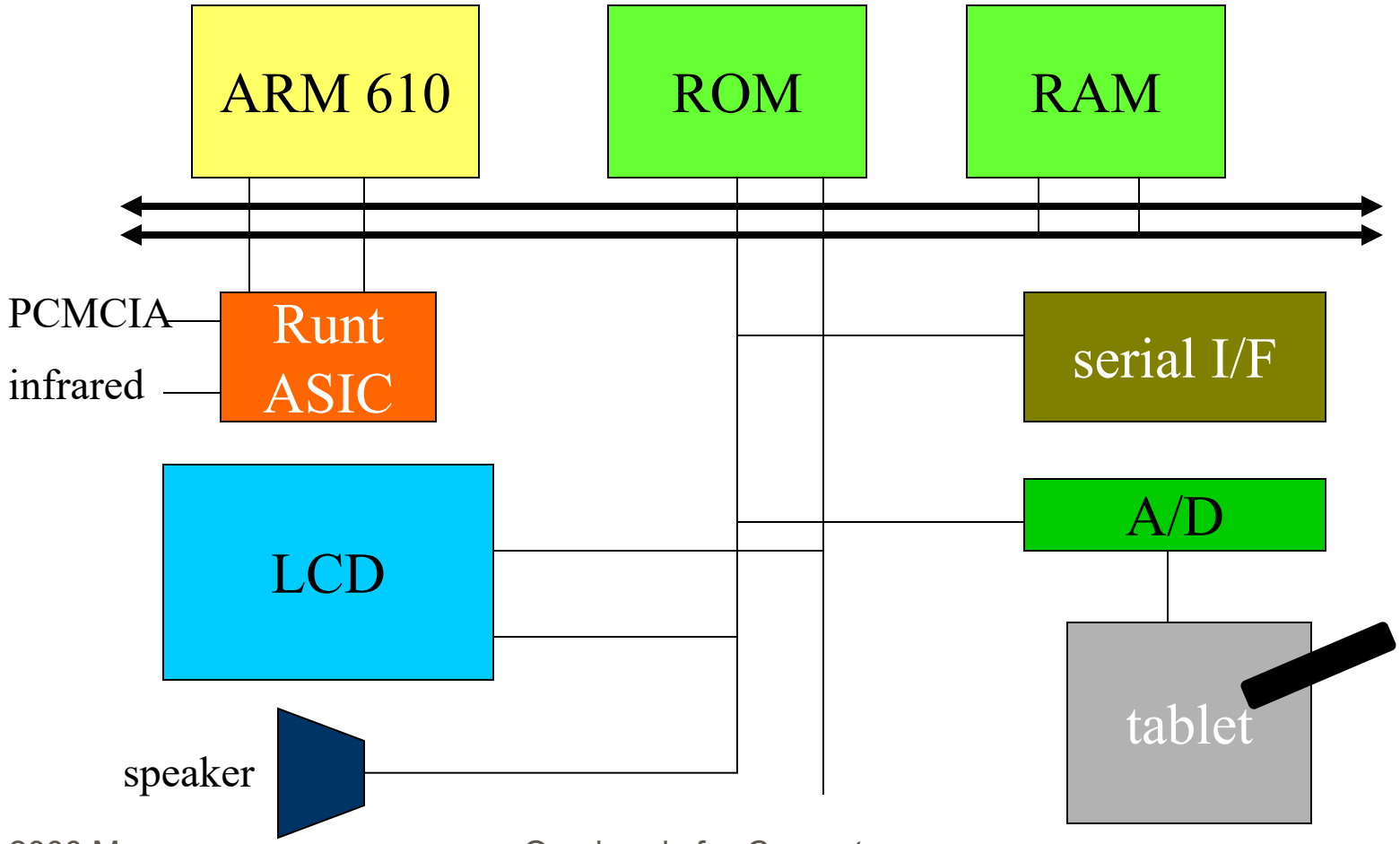
Example: StrongARM

- StrongARM system includes:
  - CPU chip (3.686 MHz clock)
  - system control module (32.768 kHz clock).
    - Real-time clock;
    - operating system timer
    - general-purpose I/O;
    - interrupt controller;
    - power manager controller;
    - reset controller.

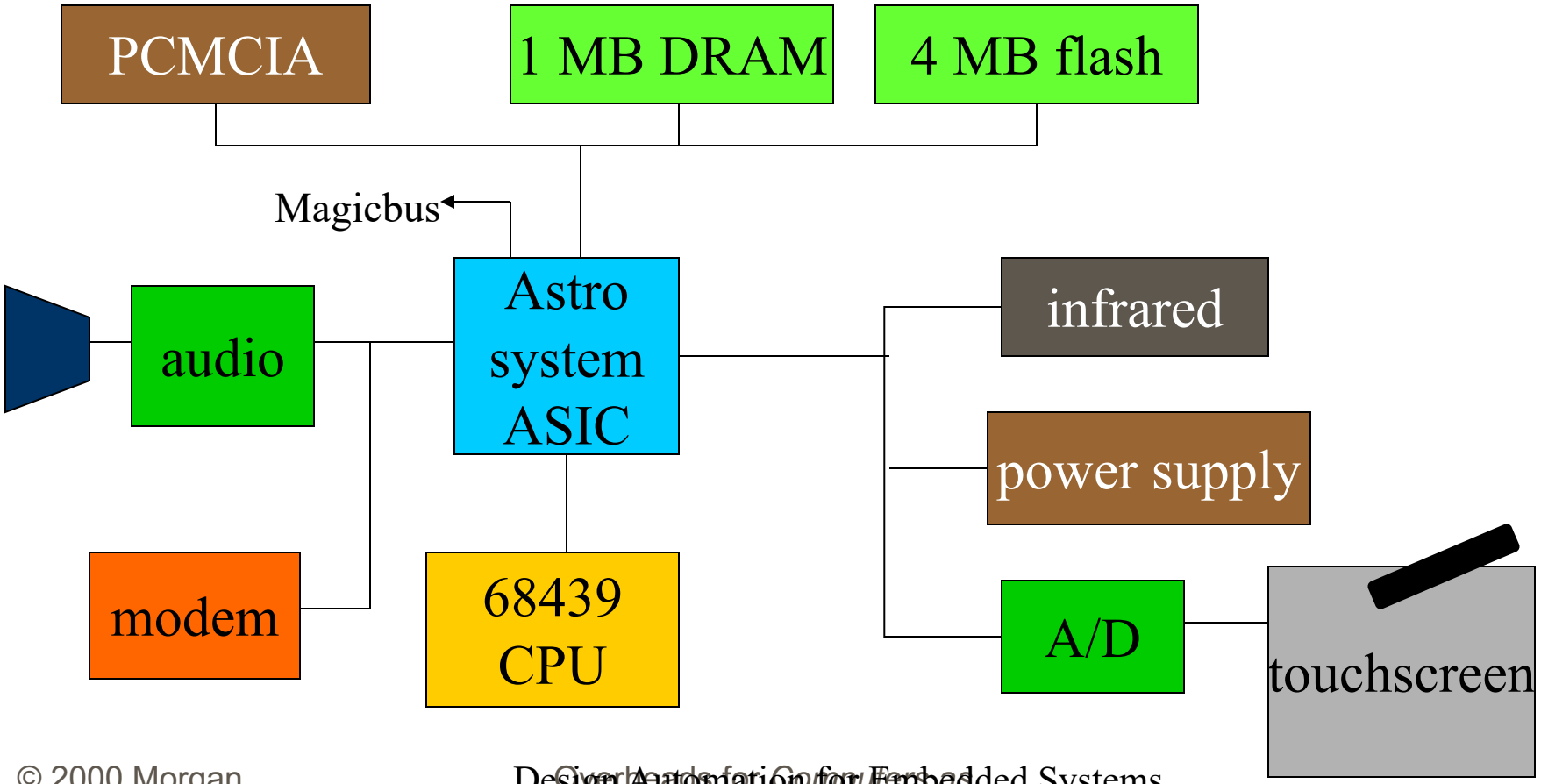
Pros and cons

- Plentiful hardware options.
- Simple programming semantics.
- Good software development environments.
- Performance-limited.

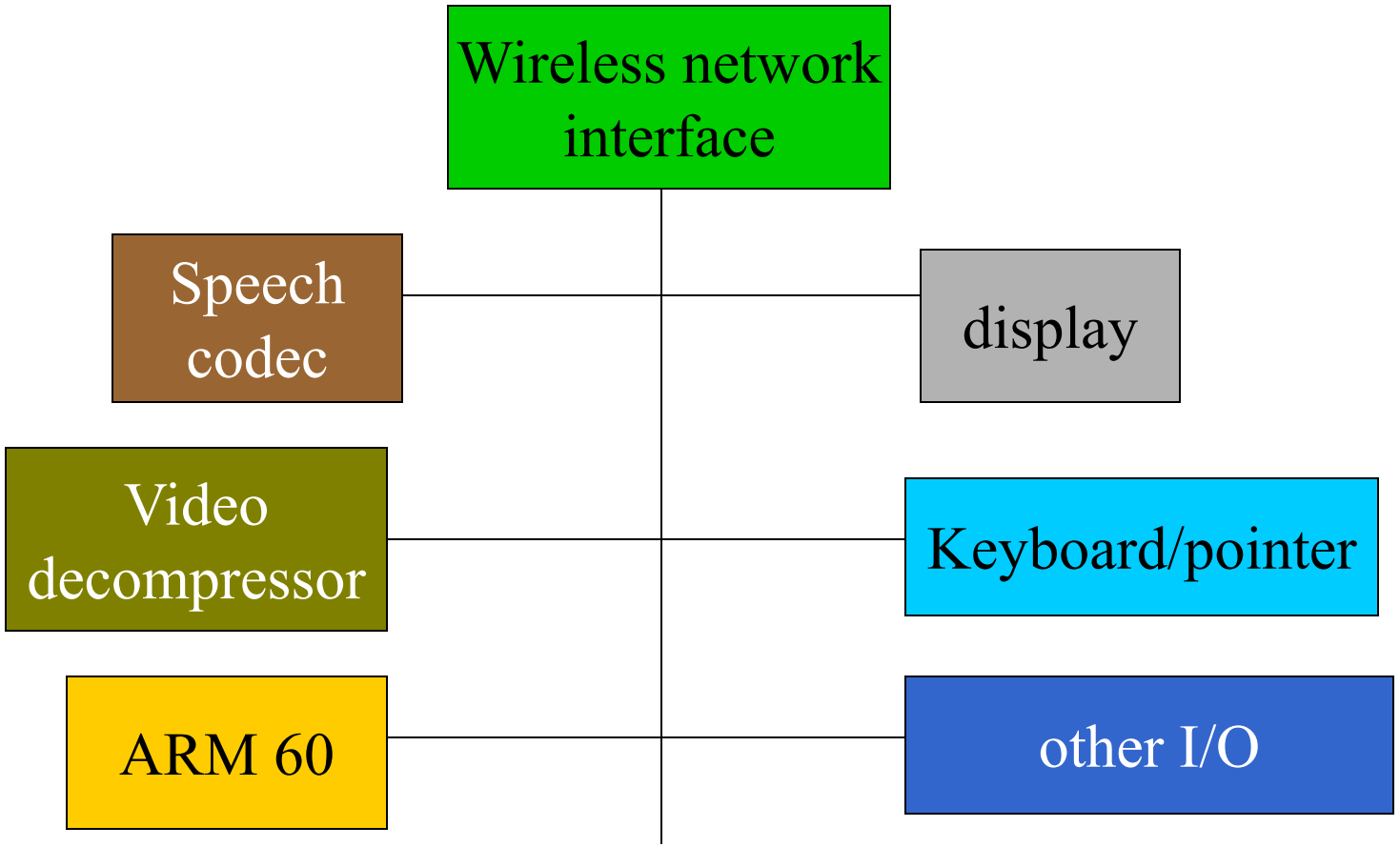
Apple Newton hardware architecture



Motorola Envoy hardware architecture



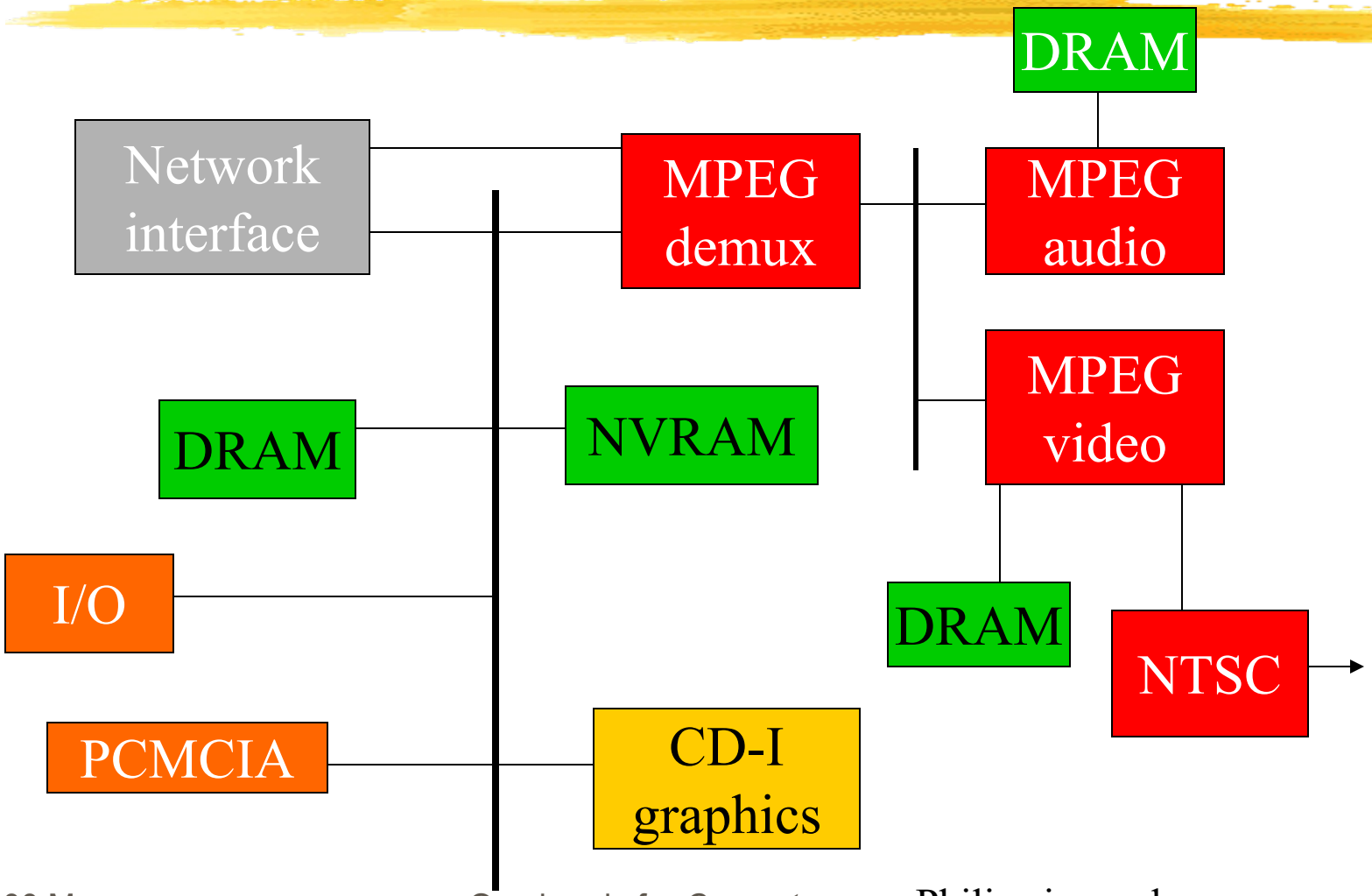
InfoPad hardware architecture



Hardware vs. software

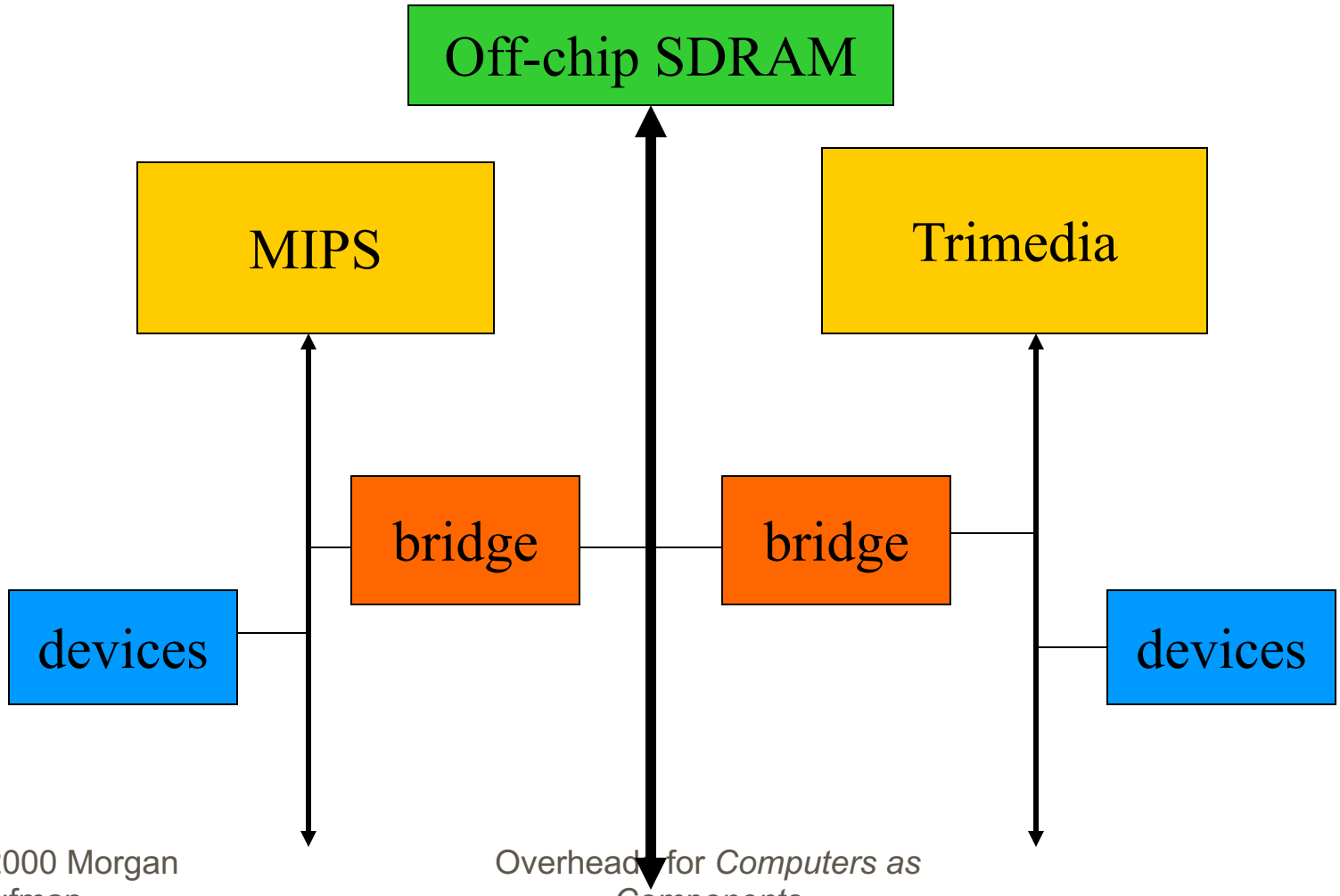
- Special-purpose hardware often consumes much less power.
- Need to think about communication between units, multiprocessing.
- Accelerators often require limits on parameters.
  - May be OK if standards limit parameters.

Philips set-top box hardware architecture



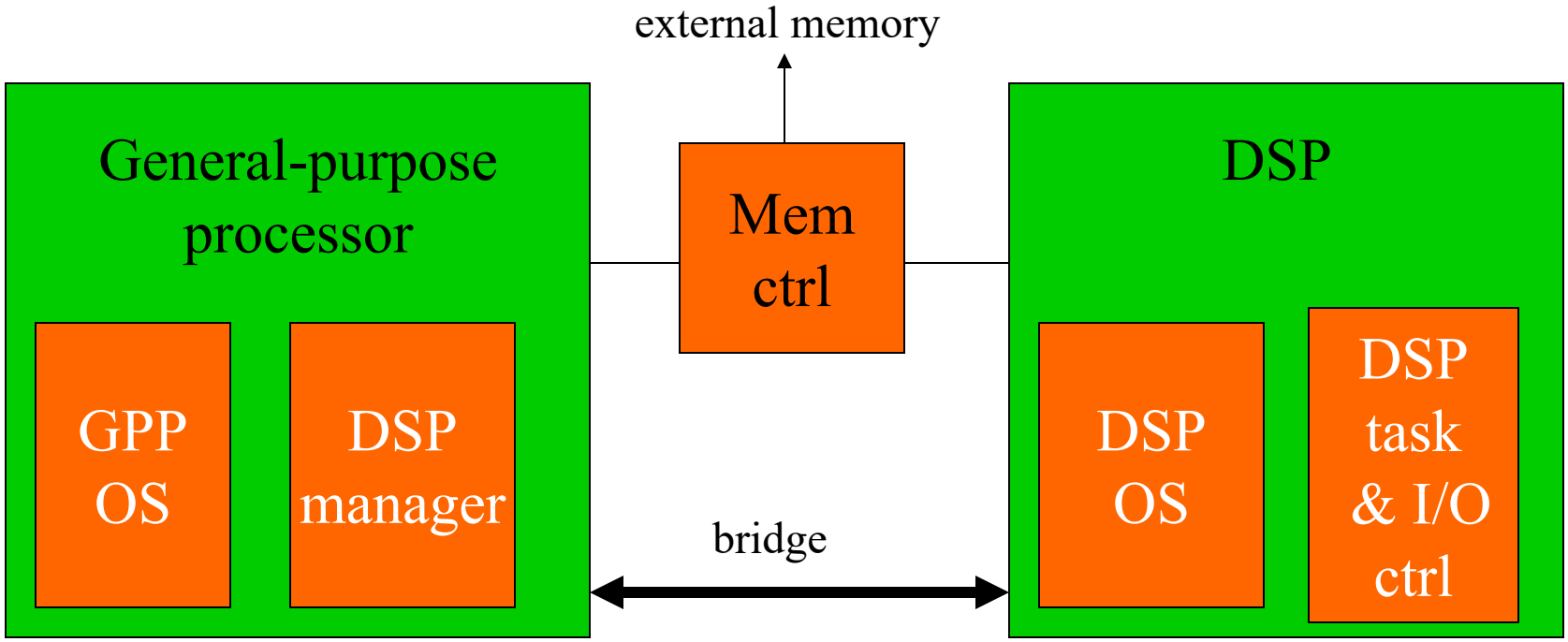
Philips journal

Viper set-top-box chip



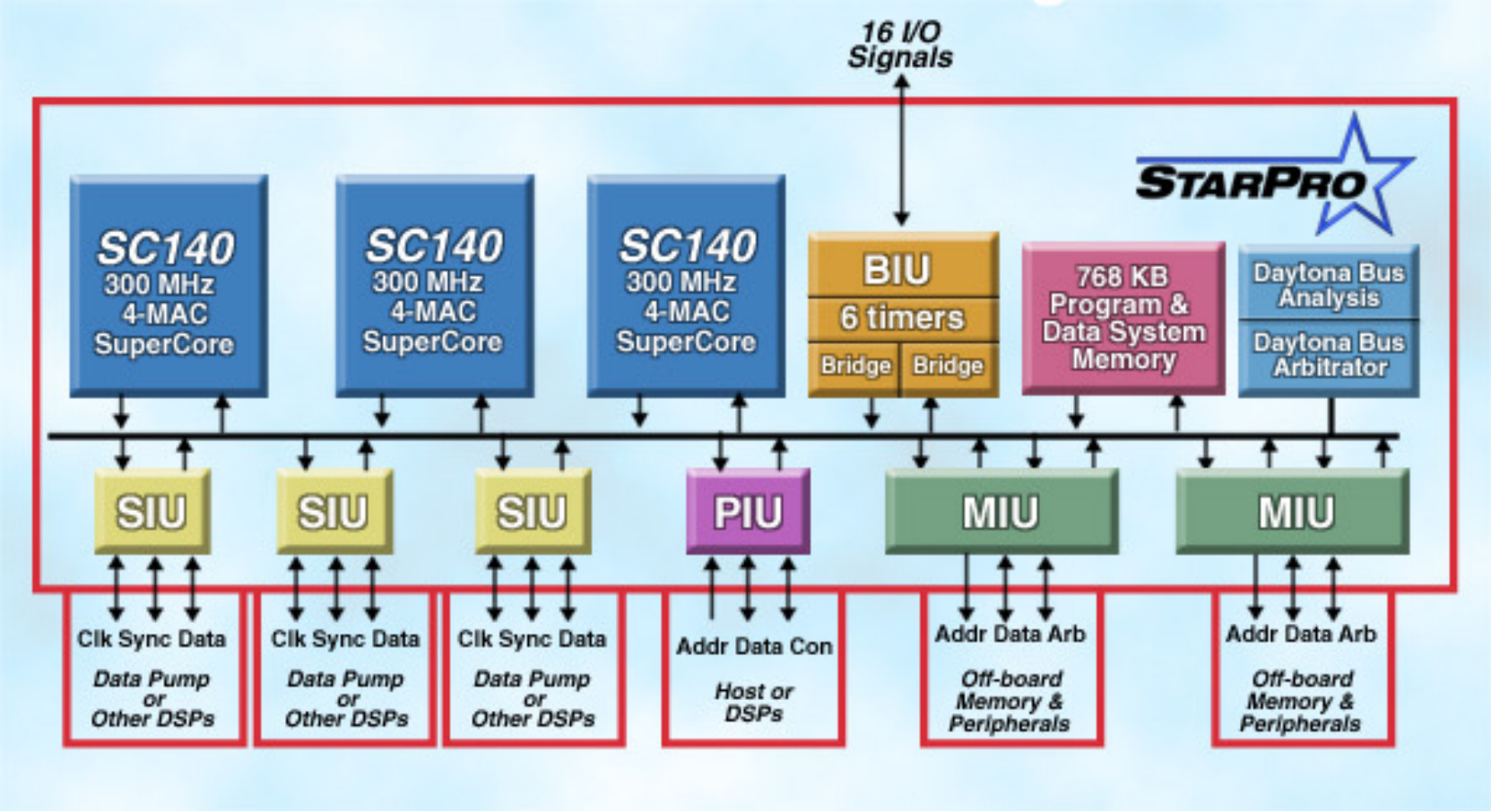
TI Open Multimedia Applications Platform

- Dual-processor shared memory system:



http://www.ti.com/sc/docs/apps/wireless/omap/overview.htm

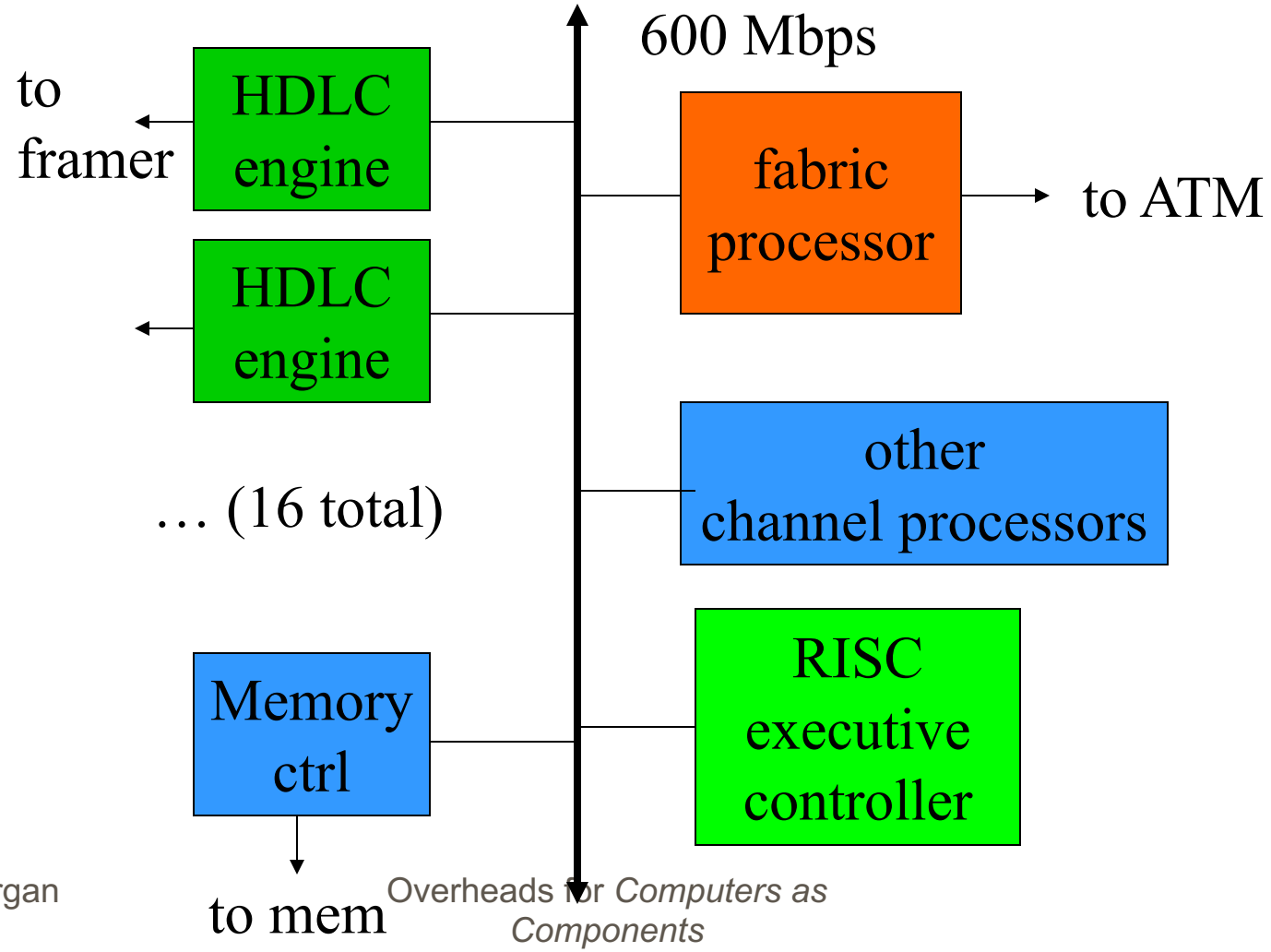
Agere StarPro platform



http://www.lucent.com/micro/starpro/arch.html

C-Port C5 network processor

http://www.cportcorp.com/products/digital.htm



Hardware and software architectures

Hardware and software are intimately related:

- software doesn't run without hardware;
- how much hardware you need is determined by the software requirements:
  - speed;
  - memory.

Software architecture

Functional description must be broken into pieces:

- division among people;
- conceptual organization;
- performance;
- testability;
- maintenance.

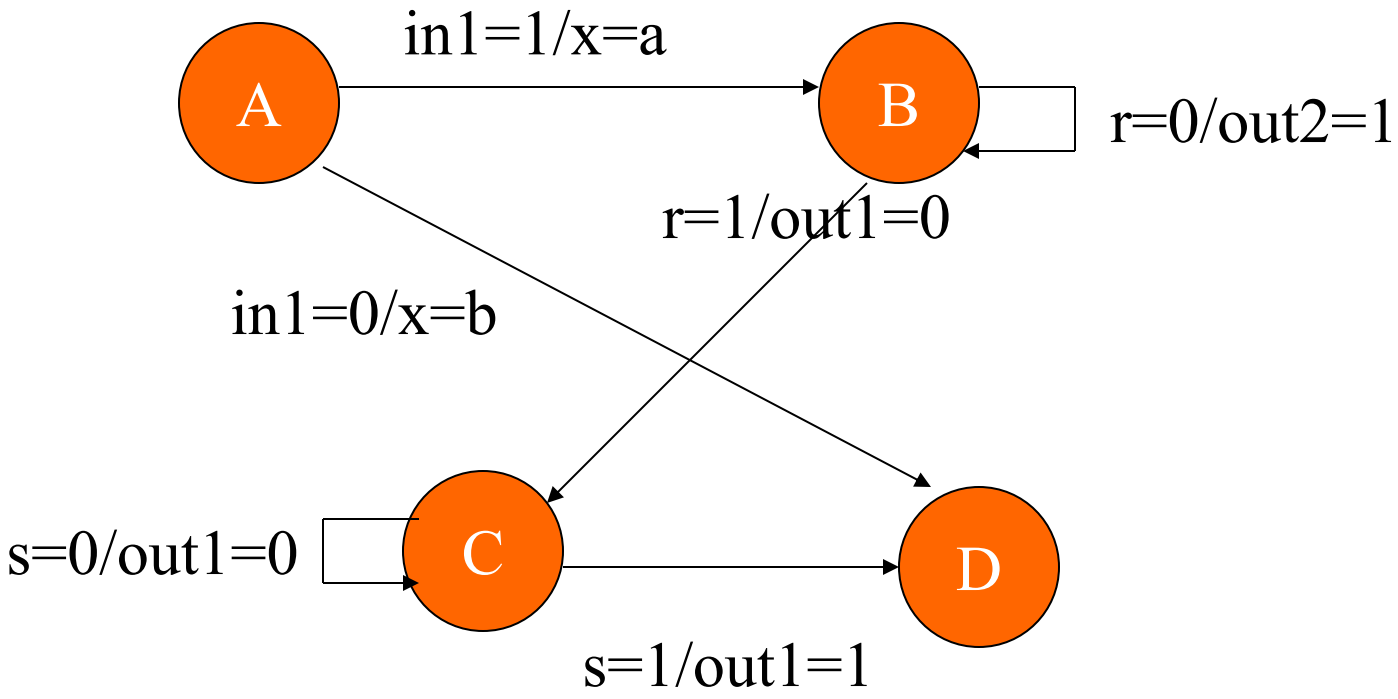
Software components

- Need to break the design up into pieces to be able to write the code.
- Some component designs come up often.
- A design pattern is a generic description of a component that can be customized and used in different circumstances.

Software state machine

- State machine keeps internal state as a variable, changes state based on inputs.
- Uses:
  - control-dominated code;
  - reactive systems.

State machine specification



C code structure

- Current state is kept in a variable.
- State table is implemented as a switch.
  - Cases define states.
  - States can test inputs.
- Switch is repeatedly evaluated in a while loop.

C state machine structure

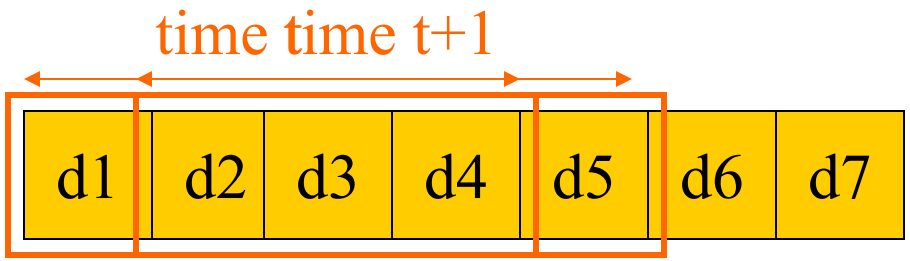
```
while (TRUE) {
  switch (state) {
     case state1: …
  }
}
```

C state table

```
switch (state) {
case A: if (in1==1) { x = a; state = B; }
  else { x = b; state = D; }
  break;
case B: if (r==0) { out2 = 1; state = B; }
  else { out1 = 0; state = C; }
  break;
case C: if (s==0) { out1 = 0; state = C; }
  else { out1 = 1; state = D; }
  break;
```

Data stream

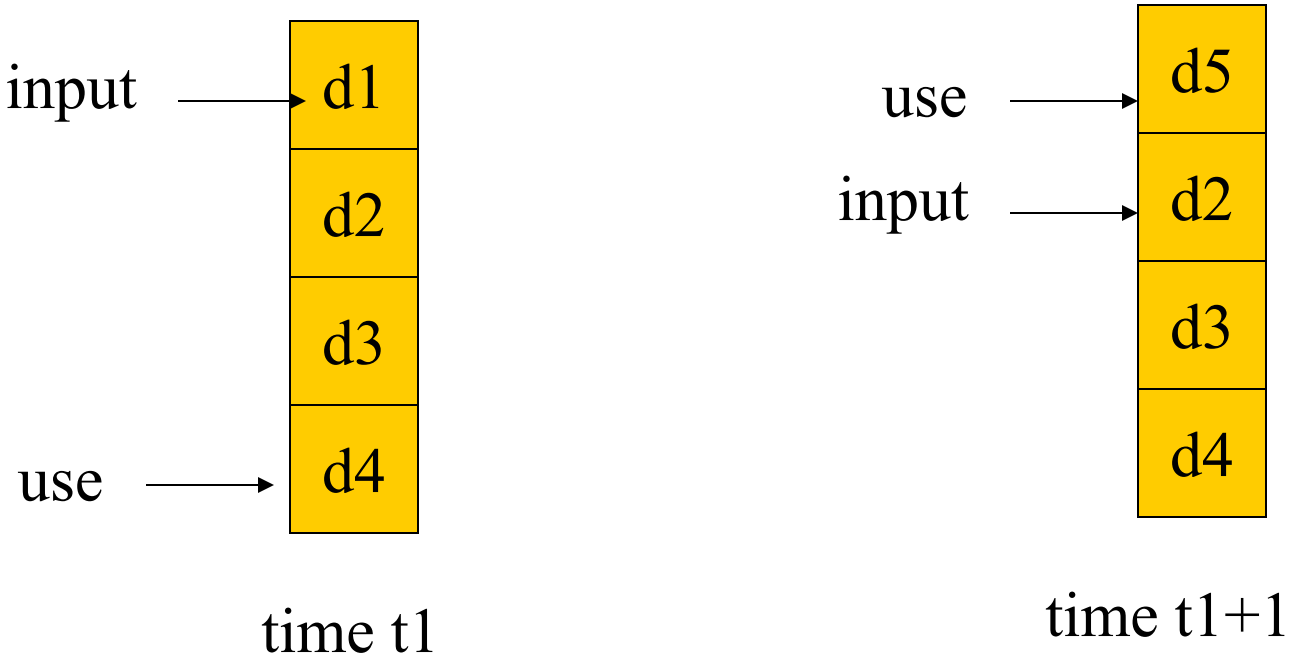
- Commonly used in signal processing:
  - new data constantly arrives;
  - each datum has a limited lifetime.



- Use a circular buffer to hold the data stream.

Circular buffers

- Indexes locate currently used data, current input data:



C circular buffer

To compute FIR filter value f:

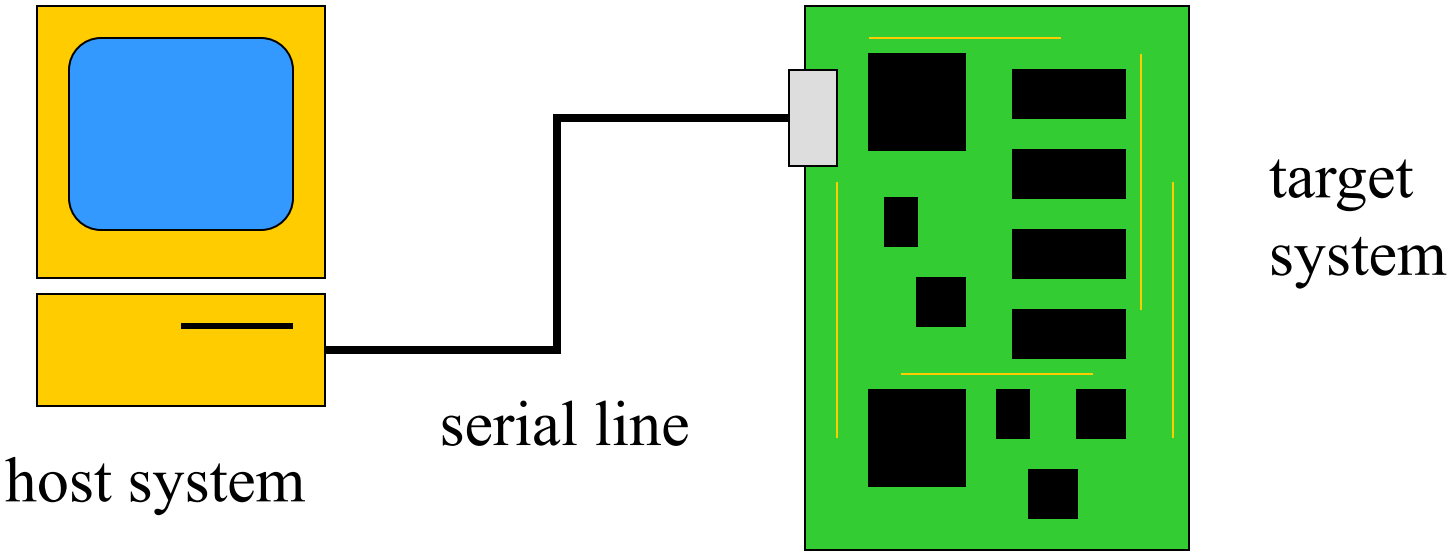
```
for (f=0, ic=0, ibuff = circ_buff_head;
  ic < N;
  ibuff = (ibuff = N-1 ? 0 : ibuff++) )
  f = f + c[ic] * circ_buff[ibuff]
```

Software design techniques

- Want to develop as much code as possible on a standard platform:
  - friendlier programming environment;
  - easier debugging.
- May need to devise software stubs to allow testing of software elements without the full hardware/software platform.

Host/target design

- Use a host system to prepare software for target system:



Host-based tools

- Cross compiler:
  - compiles code on host for target system.
- Cross debugger:
  - displays target state, allows target system to be controlled.

Evaluation boards

- Designed by CPU manufacturer or others.
- Includes CPU, memory, some I/O devices.
- May include prototyping section.
- CPU manufacturer often gives out evaluation board netlist---can be used as starting point for your custom board design.

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.

Debugging embedded systems

- Challenges:
  - target system may be hard to observe;
  - target may be hard to control;
  - may be hard to generate realistic inputs;
  - setup sequence may be complex.

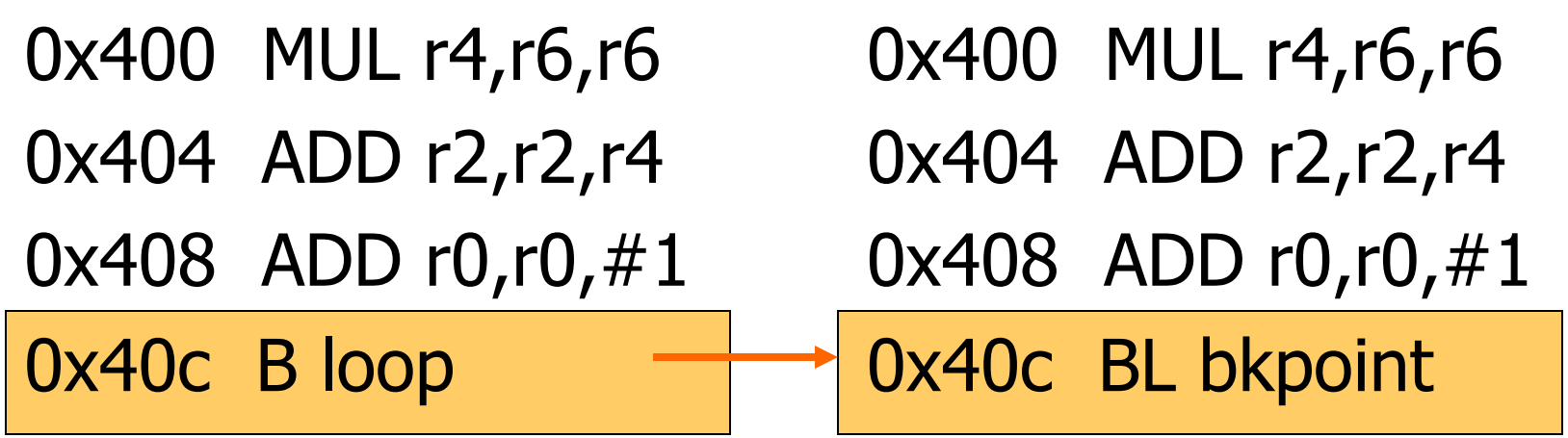
Software debuggers

- A monitor program residing on the target provides basic debugger functions.
- Debugger should have a minimal footprint in memory.
- User program must be careful not to destroy debugger program, but , should be able to recover from some damage caused by user code.

Breakpoints

- A breakpoint allows the user to stop execution, examine system state, and change state.
- Replace the breakpointed instruction with a subroutine call to the monitor program.

ARM breakpoints



Breakpoint handler actions

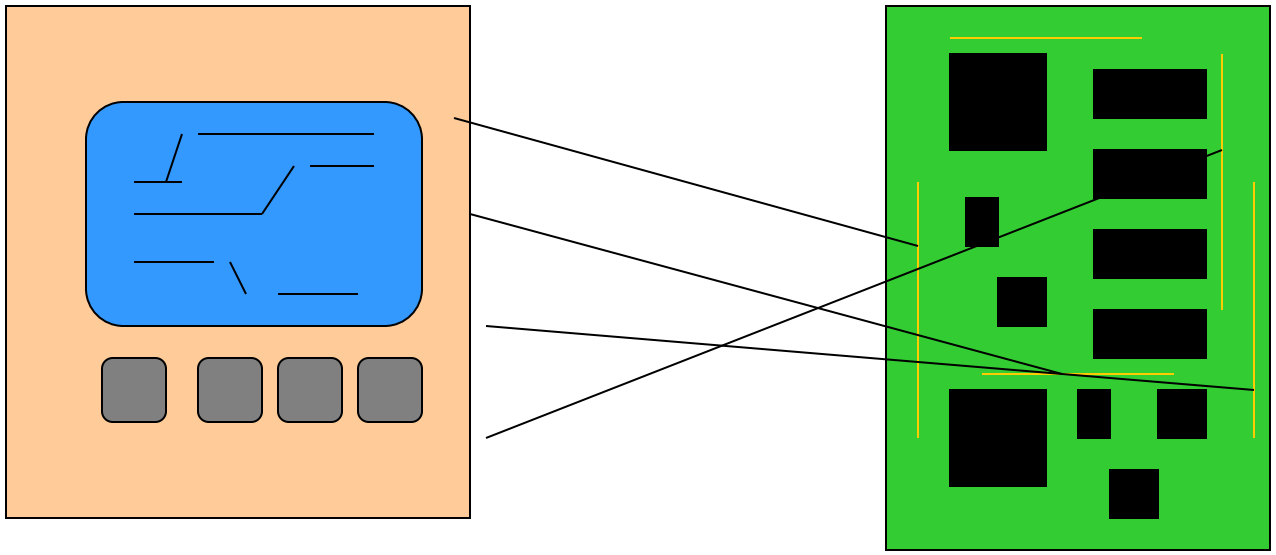
- Save registers.
- Allow user to examine machine.
- Before returning, restore system state.
  - Safest way to execute the instruction is to replace it and execute in place.
  - Put another breakpoint after the replaced breakpoint to allow restoring the original breakpoint.

In-circuit emulators

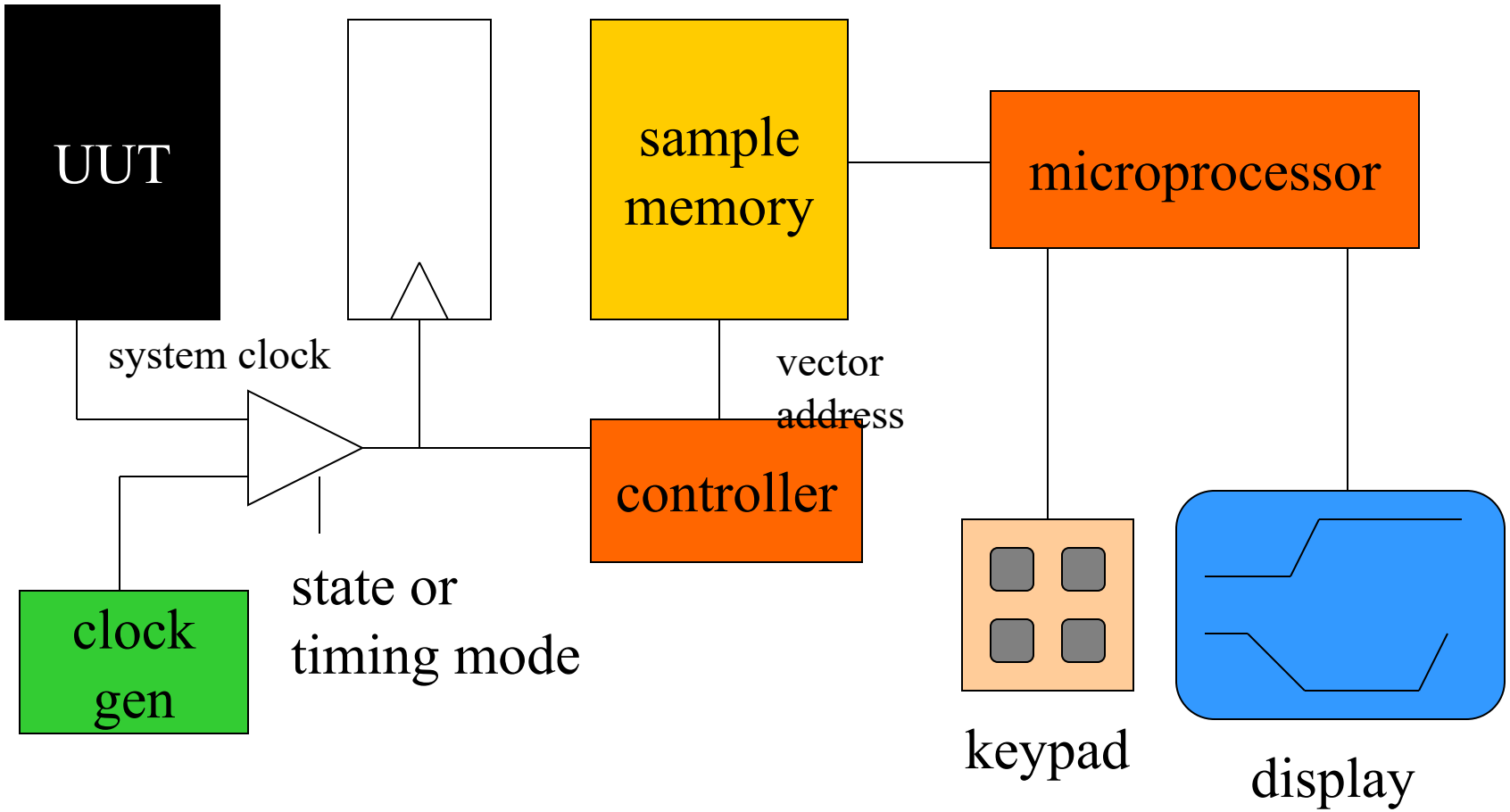
- A microprocessor in-circuit emulator is a specially-instrumented microprocessor.
- Allows you to stop execution, examine CPU state, modify registers.

Logic analyzers

- A logic analyzer is an array of low-grade oscilloscopes:



Logic analyzer architecture



How to exercise code

- Run on host system.
- Run on target system.
- Run in instruction-level simulator.
- Run on cycle-accurate simulator.
- Run in hardware/software co-simulation environment.

Manufacturing testing

- Goal: ensure that manufacturing produces defect-free copies of the design.
- Can test by comparing unit being tested to the expected behavior.
  - But running tests is expensive.
- Maximize confidence while minimizing testing cost.

Testing concepts

- Yield: proportion of manufactured systems that work.
  - Proper manufacturing maximizes yield.
  - Proper testing accurately estimates yield.
- Field return: defective unit that leaves the factory.

Faults

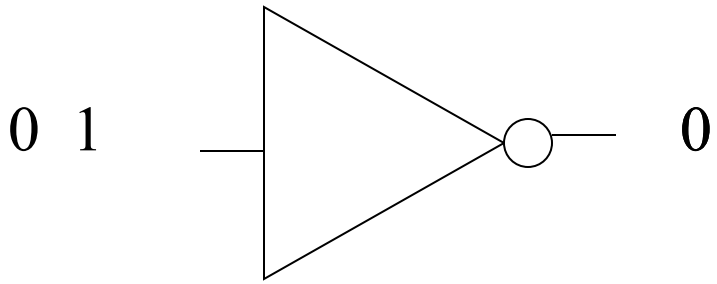
- Manufacturing problems can be caused by many thing.
- Fault model: model that predicts effects of a particular type of fault.
- Fault coverage: proportion of possible faults found by a set of test.
  - Having a fault model allows us to determine fault coverage.

Software vs. hardware testing

- When testing code, we have no fault model.
  - We verify the implementation, not the manufacturing.
  - Simple tests (e.g., ECC) work well to verify software manufacturing.
- Hardware requires manufacturing tests in addition to implementation verification.

Hardware fault models

- Stuck-at 0/1 fault model:
  - output of gate is always 0/1.



Combinational testing

- Every gate can be stuck-at-0, stuck-at-1.
- Usually test for single stuck-at-faults.
  - One fault at a time.
  - Multiple faults can mask each other.
- We can generate a test for a gate by:
  - controlling the gate's input;
  - observing the gate's output through other gates.

Sequential testing

- A state machine is combinational logic + registers.
- Sequential testing is considerably harder.
  - A single stuck-at fault affects the machine on every cycle.
  - Fault behavior on one cycle can be masked by same fault on other cycles.

Scan chains

- A scannable register operates in two modes:
  - normal;
  - scan---forms an element in a shift register.
- Using scan chains reduces sequential testing to combinational testing.
  - Unloading/unloading scan chain is slow.
  - May use partial scan.

Test generation

- Automatic test pattern generation (ATPG) programs: produce a set of tests given the logic structure.
- Some faults may not be testable---redundant.
  - Timeout on a fault may mean hard-to-test or untestable.

Boundary scan

- Simplifies testing of multiple chips on a board.
  - Registers on pins can be configured as a scan chain.