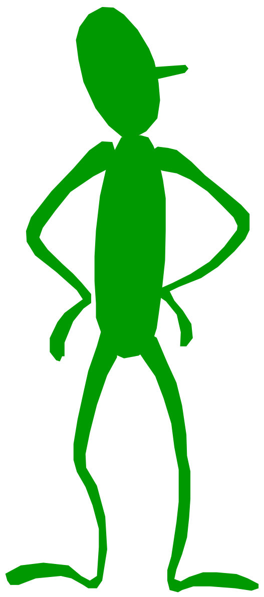
VLSI Systems Design

Design Project: Practical Aspects

I am a VHDL expert.
But how applying
in real live – for my MP3 player!



Overview

applying the "description-synthesis" design method in practice

Goal: You are able to master your own VHDL project. You have basic notions about HW/SW co-design.

Project Goal

- Goal:

  design of an electronic system from specification down to ASIC/FPGA

- Problem:

  one of the most difficult tasks in a VLSI project design is to find the starting design point

- Basic Steps:

  in order to proceed in a structured manner, you should perform the following steps

  - block diagram
  - HW/SW co-design (hardware/software co-design)
  - IP cores (intellectual property cores)

↙ *hardware* *software* ↘ *co-design*

- FSMD architecture model
- VHDL coding & simulation

- structured software design
- C coding, compiling

↘ *hardware* *software* ↙ *co-design*

- hardware/software system simulation
- synthesis, place & route
- back-annotation & simulation (formal design verification)

Initial System Design Steps

- System design steps

block diagram

1. identify your chip in the overall system
2. define the chip IO and group them to blocks
3. identify functional units of your chip
4. identify the interconnection between your units

HW/SW co-design

5. identify speed sensitive (HW) and control sensitive (SW) tasks
6. define the "intelligence" of each functional unit

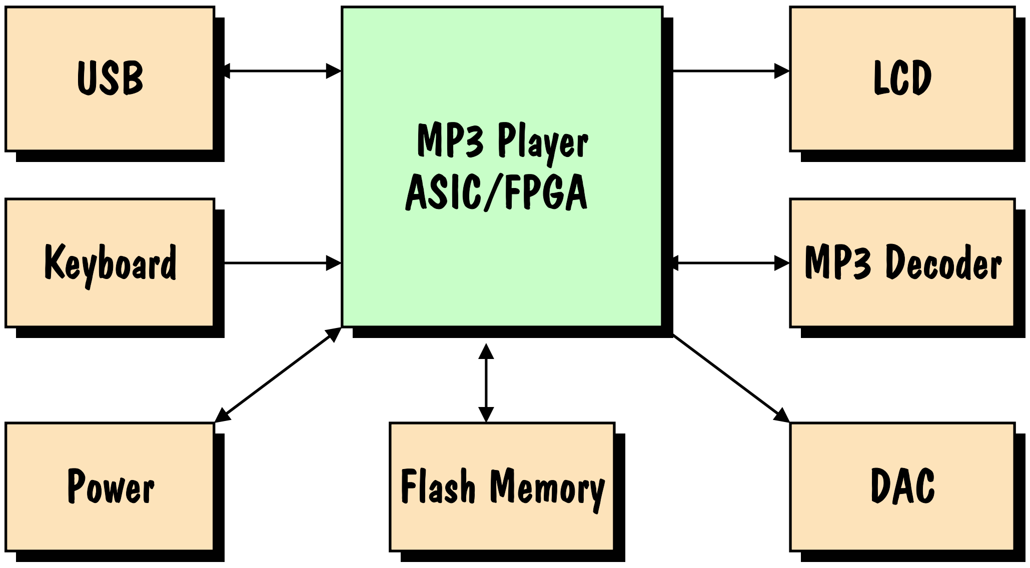
IP cores

7. identify IP cores
8. organize as much as possible IP cores (tools, core generators, old designs, internet)
9. update design if necessary according to available IP cores
10. define inter-process communication
11. define the interconnections between your units

- In the classical HW/SW co-design approach, the design process is continued as long as possible independent of its implementation. HW/SW design units are identified at the very end of the design steps. In smaller designs, as it is in our case, the HW/SW co-design step is done in an early phase.

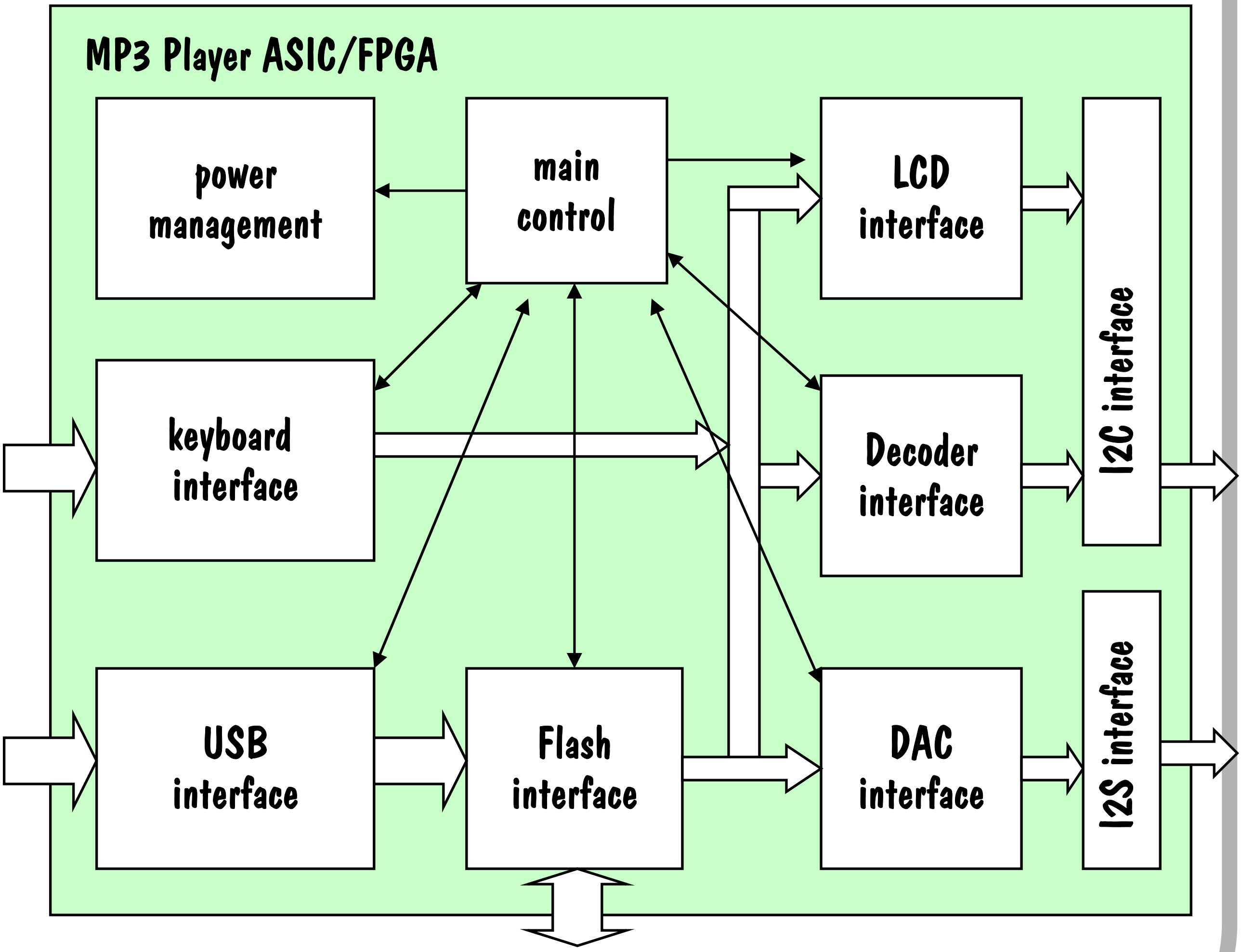
Project MP3 Player: step 1 (block diagram)

- Step 1: identify your chip in the overall system



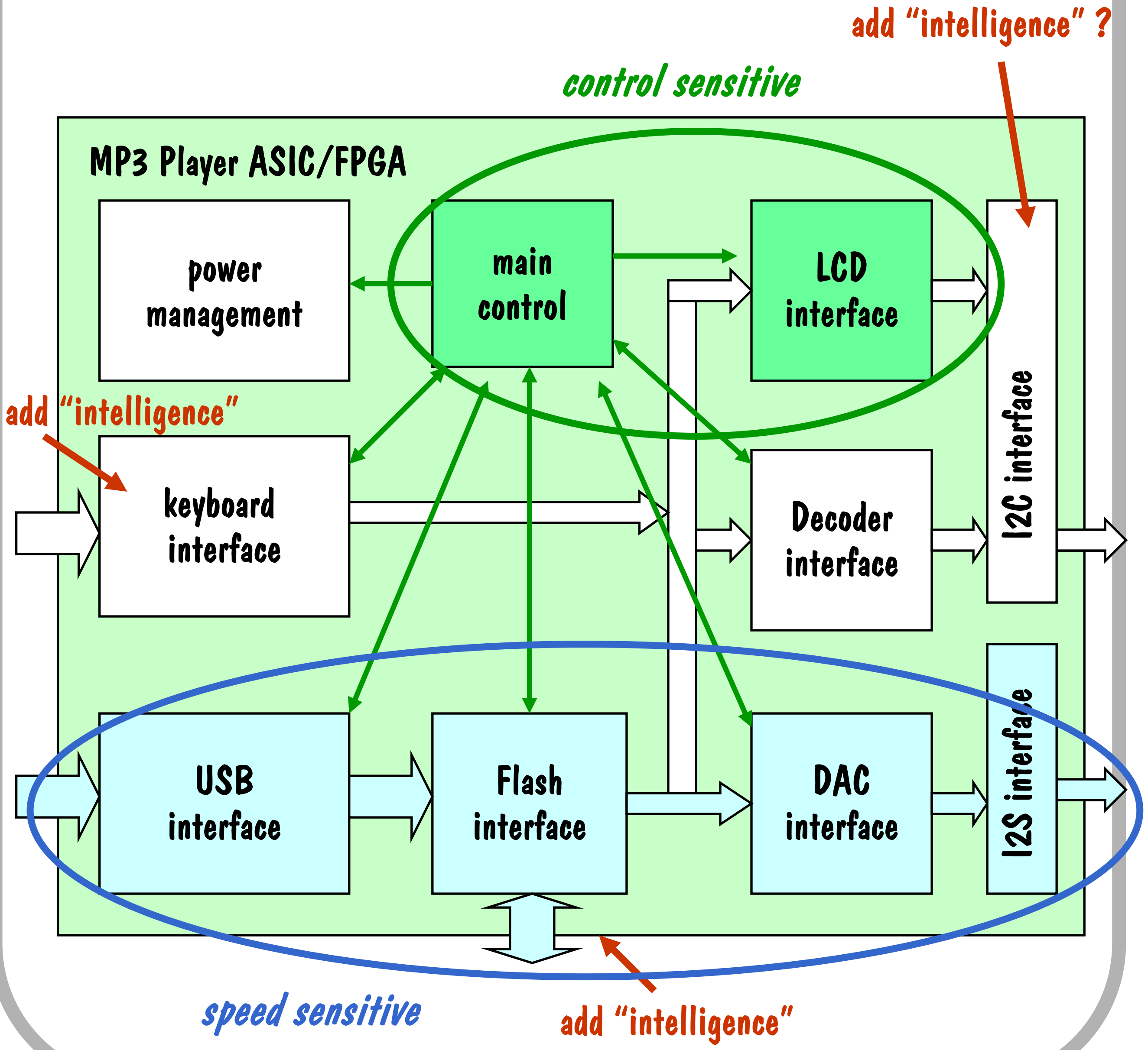
Project MP3 Player: step 2-4 (block diagram)

- Step 2: define the chip IO and group them to blocks
- Step 3: identify functional units of your chip
- Step 4: find the interconnections between your units



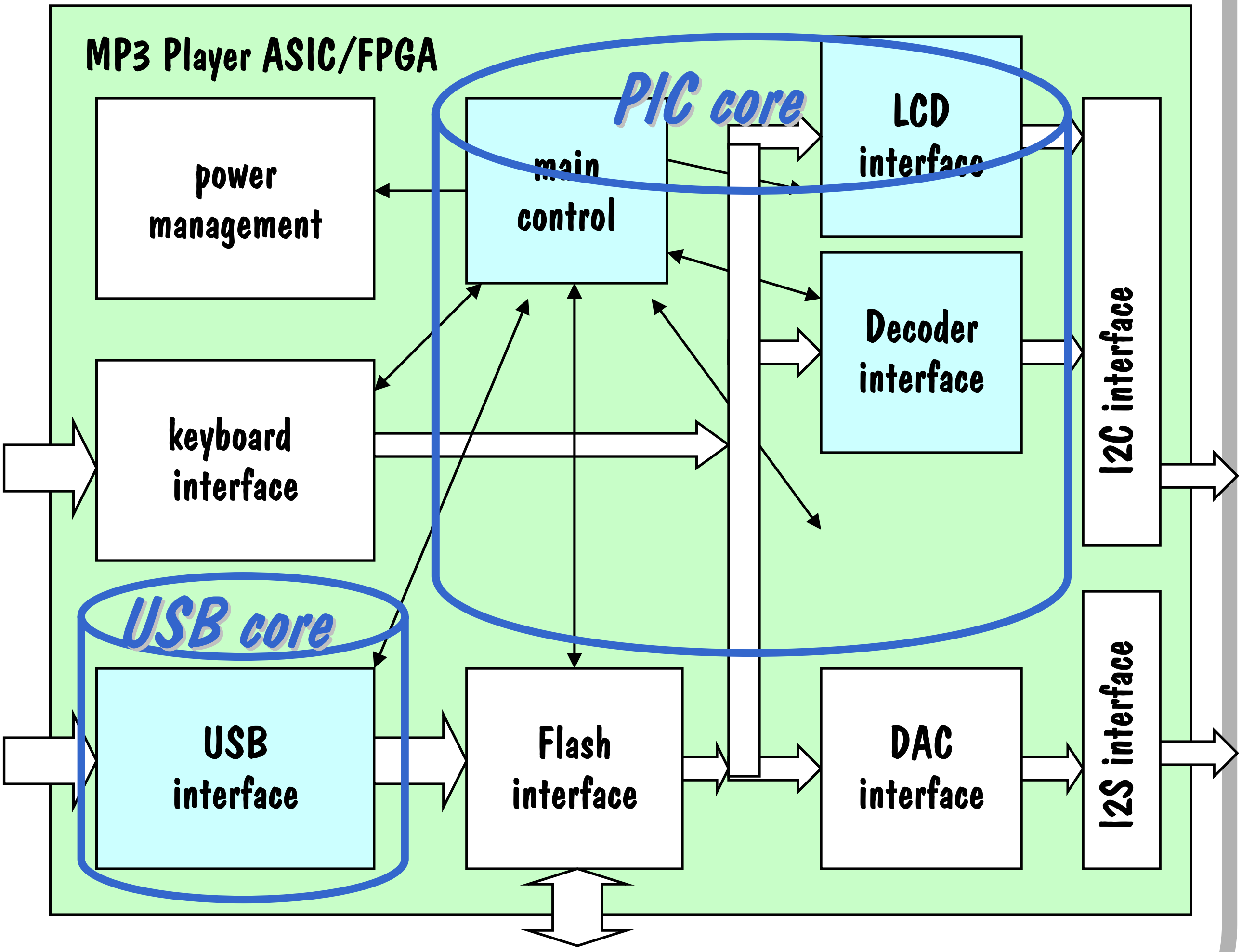
Project MP3 Player: step 5 (HW/SW Co-Design)

- Step 5: identify speed and control sensitive tasks
- Step 6: define the "intelligence" of each functional unit



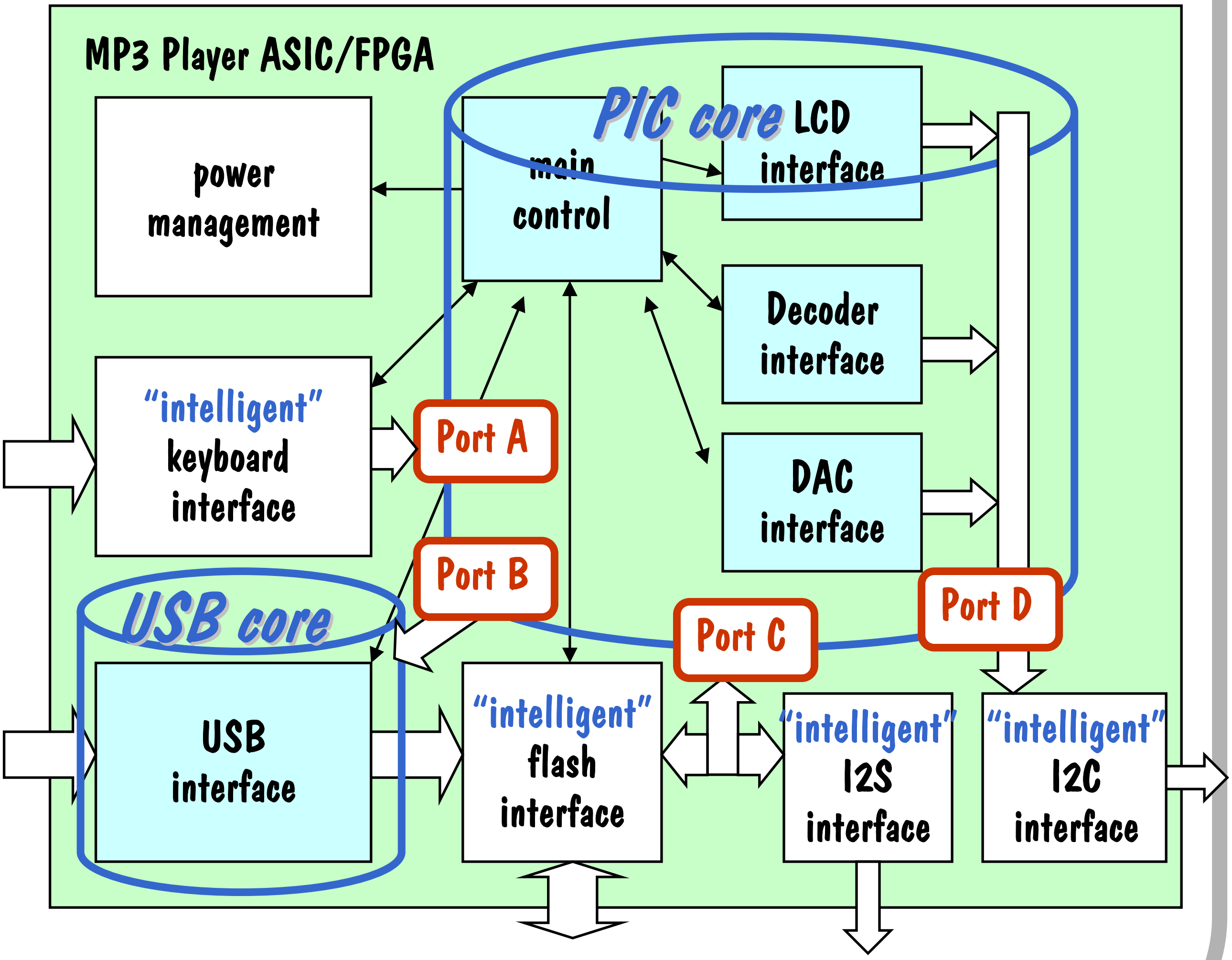
Project MP3 Player: step 7-8 (Hardware Design)

- Step 7: identify IP cores
- Step 8: organize as much as possible IP cores (tools, core generator, old designs, internet)



Project MP3 Player: step 9-11 (Hardware Design)

- Step 9: update design if necessary according to available IP cores
- Step 10: define inter-process communication
- Step 11: define the interconnection between units



Hardware/Software Design Steps

- Hardware design project steps:

FSMD architecture model

I. imagine your chip working in the target system, identify and describe its basic functional units in a data-flow view

II. find the RTL structure of each of the above data-flow functions and update your block diagram by allocating your RTL structure to one or more functional units

III. fix in detail the operation of your functional units (local intelligence or data-path only) and add FSMs if required, fix the detailed interconnections between your units

IV. design all FSMs, define clock strategy, use colored data-flow, be careful with the inter-process communications

VHDL coding

V. VHDL coding of your RTL design

VI. test bench design

VII. simulate your VHDL design with test bench

- Software design project steps:

structured software design

I. design the software structure as learned in SW engineering courses

II. define the data structure

III. define the HW/SW communication

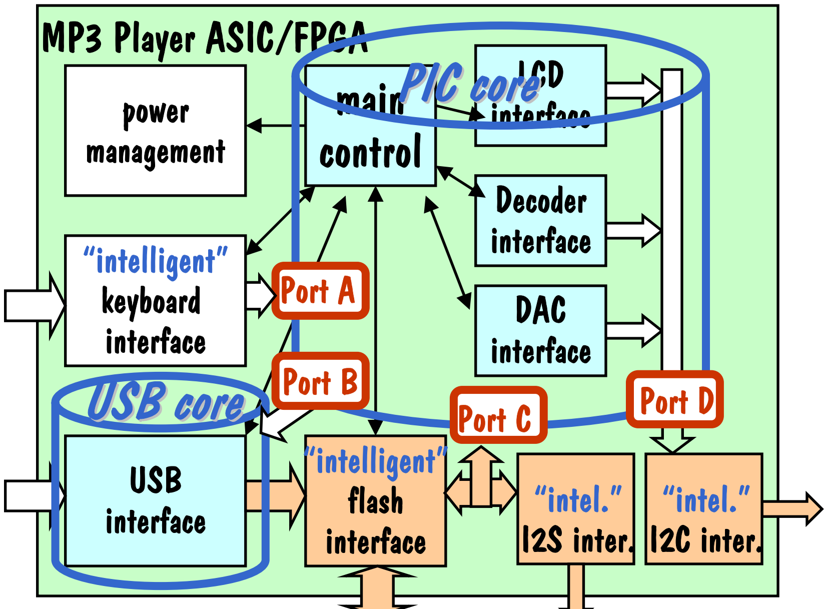
C coding

IV. develop the C code

V. compile & verify your C code

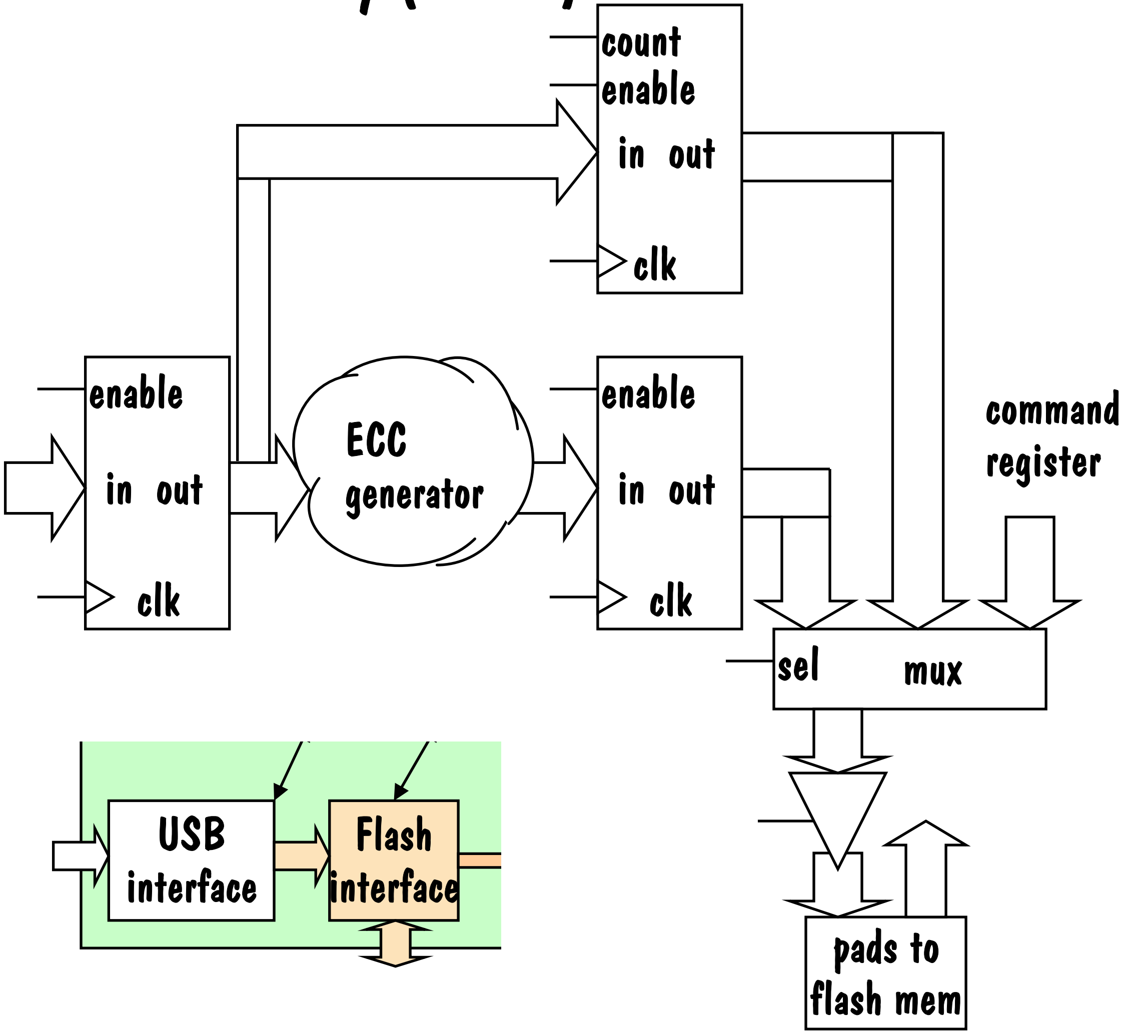
Project MP3 Player: step 1 (Hardware design project steps)

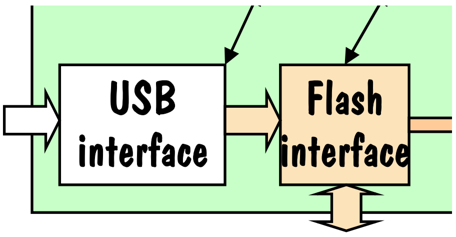
- Step 1: imagine your chip working in the target system, identify and describe its basic functional units in a data-flow view
  - download MP3 song from host to flash memory (flow 1):
    - ✓ generate flash command, generate flash address
    - ✓ load byte from USB into register
    - ✓ use byte to execute ECC (Hamming code)
    - ✓ update flash address
    - ✓ store byte into flash
    - ✓ write ECC code after 512 bytes
    - ✓ generate write-to-flash after 512 bytes
    - ✓ use pipeline structure to speed up data transfer



Project MP3 Player: step II (hardware design project steps)

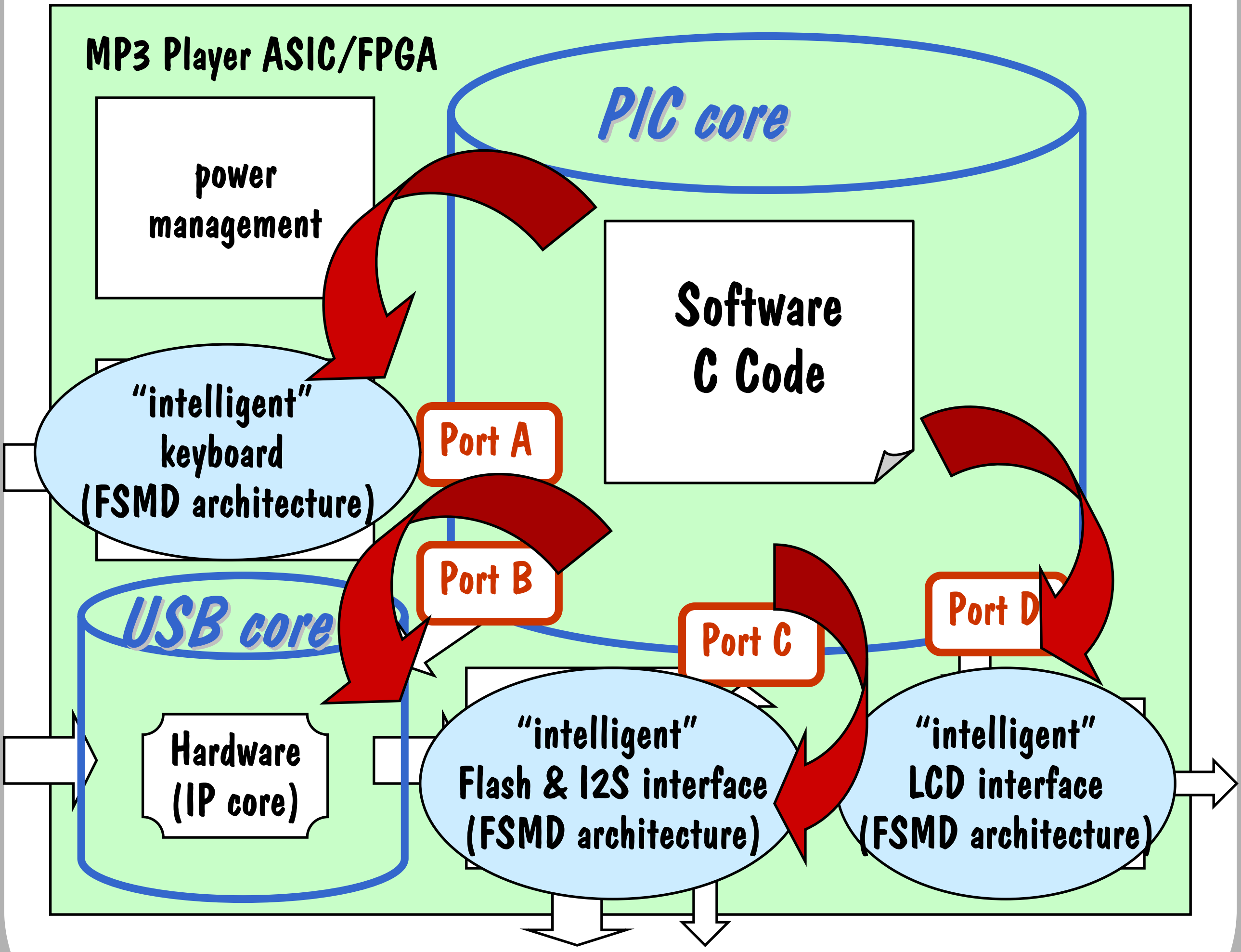
- Step II: find the RTL structure of each of the previous data-flow functions and update your block diagram by allocating your RTL structure to one or more functional units
  - download MP3 song from host to flash memory (flow 1):





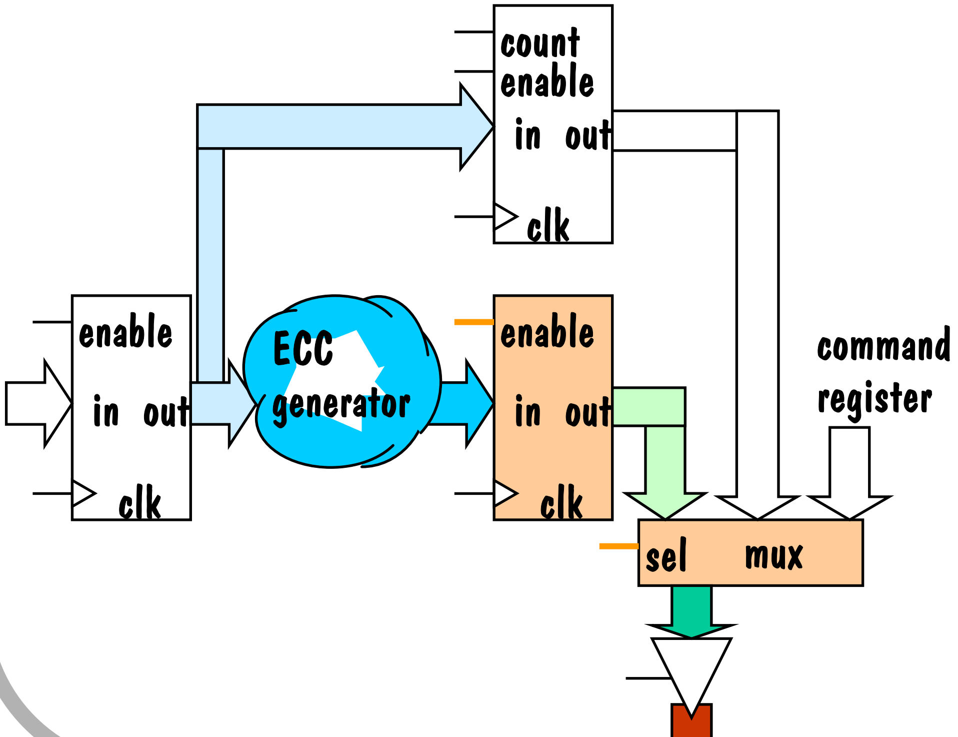
Project MP3 Player: step III (hardware design project steps)

- Step III: fix in detail the function of your functional units (local intelligence or data-path only) and add FSMs if required, fix the detailed interconnections between your units


MP3 Player ASIC/FPGA
PIC core
power management
Software C Code
"intelligent" keyboard (FSMD architecture)
Port A
Port B
Port C
Port D
USB core
Hardware (IP core)
"intelligent" Flash & I2S interface (FSMD architecture)
"intelligent" LCD interface (FSMD architecture)

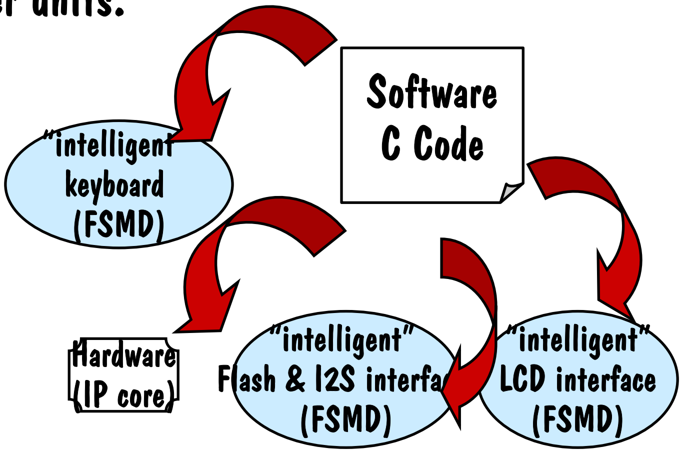
Project MP3 Player: step IVa

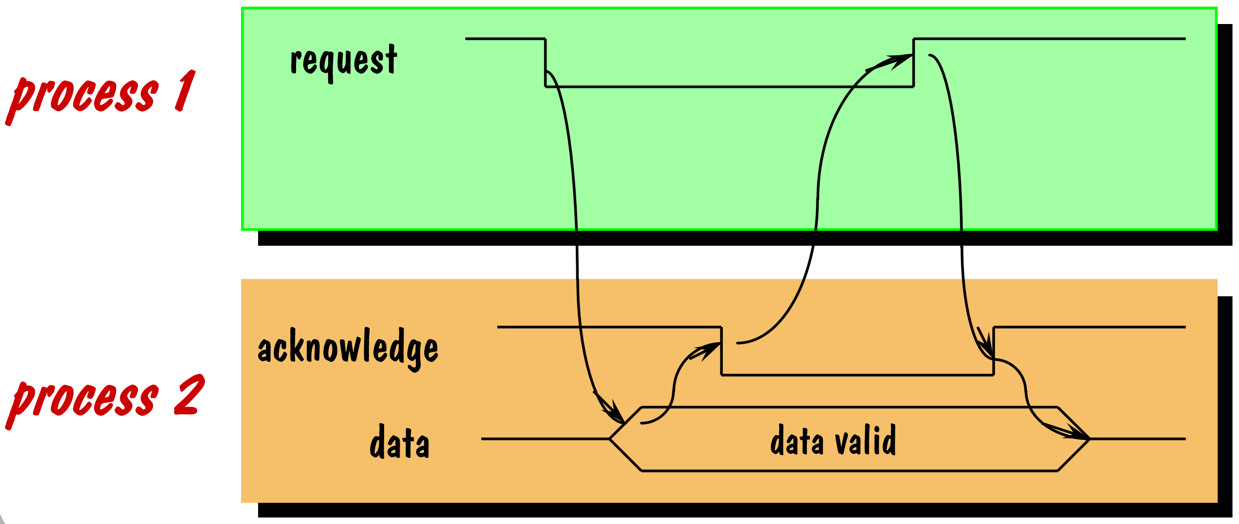
- Step IVa: design all FSMs, define clock strategy, use colored data-flow, be careful with the inter-process communications
  - Clock strategy: Rising edge for data-paths, falling edge for IP cores and FSMs. All handshake signals between FSMDs and IP cores on falling edge.
  - Colors: make a lot of copies of your RTL data path
  - Colors: for each data-flow step, color the old active data paths leaving a register blue, the new active data-paths leaving a register green, and data-paths treated with a combinatorial function in the corresponding dark color. Active control signals and its blocks are orange. All other data-signals are red. Red signals are dominant. Be sure that no red signals enter a FSM, and no darkened or red signals attack asynchronous set/reset of FFs.



Project MP3 Player: step IVb

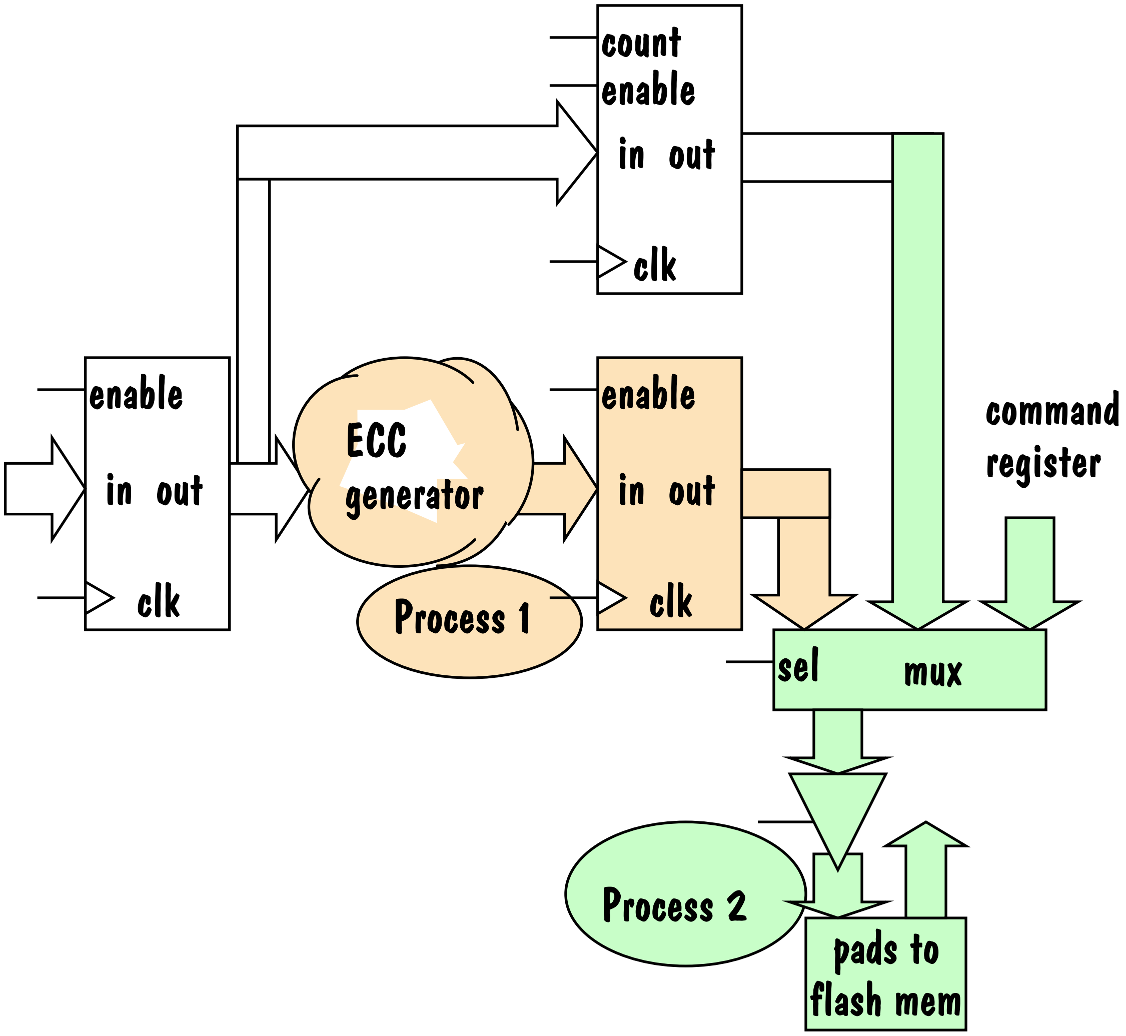
- Step IVb: design all FSMs, define clock strategy, use colored data-flow, be careful with the inter-process communications
  - we decide to use 3 different FSMs in addition to the ones present in IP cores
  - the PIC processor core is the main unit, which communicates with all other FSMD or core units, thus use inter-process communication. There is no communication in-between the other units.


Software
C Code
"intelligent" keyboard (FSMD)
Hardware (IP core)
"intelligent" Flash & I2S interface (FSMD)
"intelligent" LCD interface (FSMD)



Project MP3 Player: step V

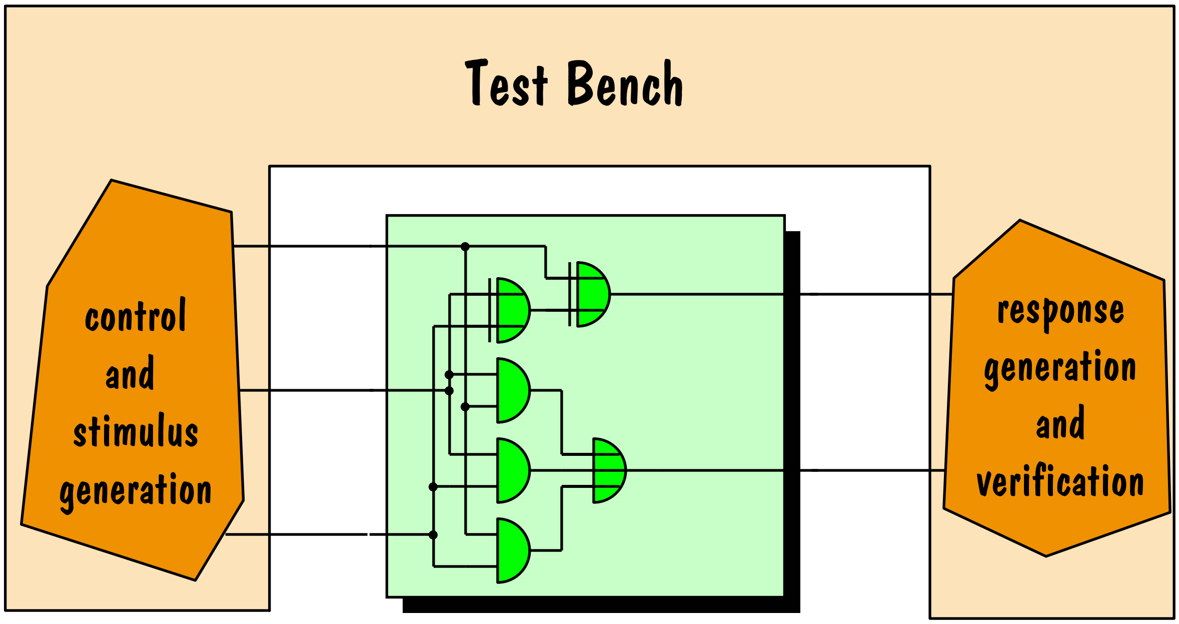
- Step V: VHDL coding of your RTL design
  - use a processes for data-path manipulation and its succeeding register
  - use 2 processes for a FSM:
    - one process for transition table (VHDL case)
    - one process for next state (state register)
    - continuous assignment for output function



Project MP3 Player: step VI

- Step VI: test bench design

  the design of a test bench is one of the most time consuming and important tasks. A test bench will be re-used several times during the different design steps as well as for chip test (have a look at vlsi21)


Test Bench
control and stimulus generation
response generation and verification

device under test (DUT)

Final System Design Steps

- Hardware design project steps:

system simulation

12. system test bench design
13. hardware/software system simulation with test bench

synthesis place and route

14. synthesis of logic level design
15. simulation of logic level with test bench
16. place & route your design for target technology

verify

17. back annotation and simulation with test bench
18. (formal design verification)

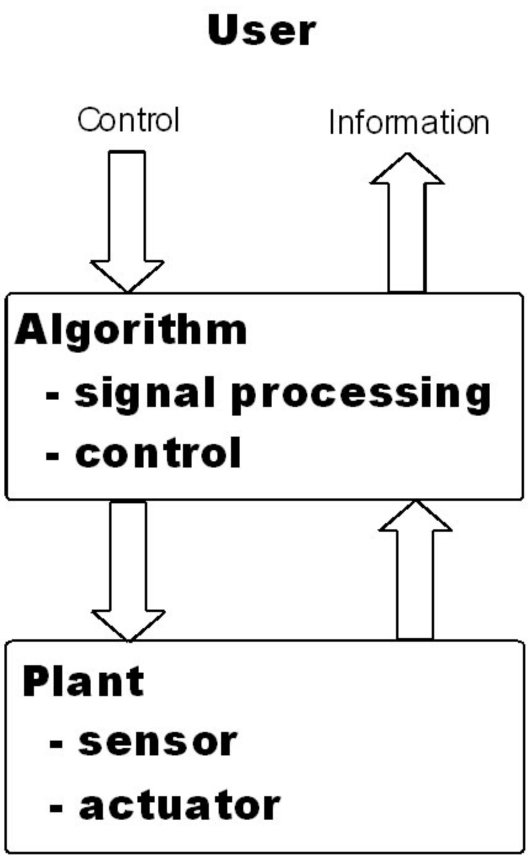
19. chip fabrication

test

20. chip test with test bench
21. in system test

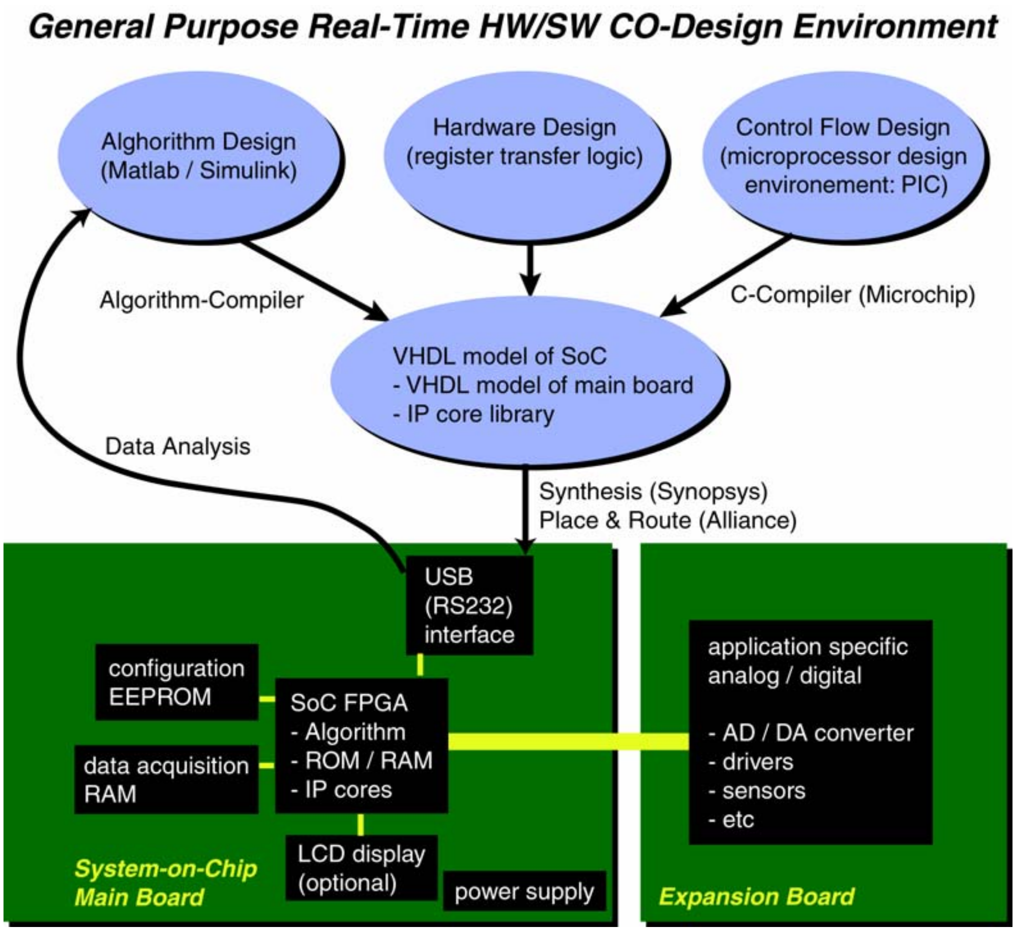
Block Diagram of a General System

- A general system is composed of three elements:
  - user
  - algorithm
  - plant
- all three items interact with each other resulting in 2 closed loops
- The closed loops may have real-time constraints



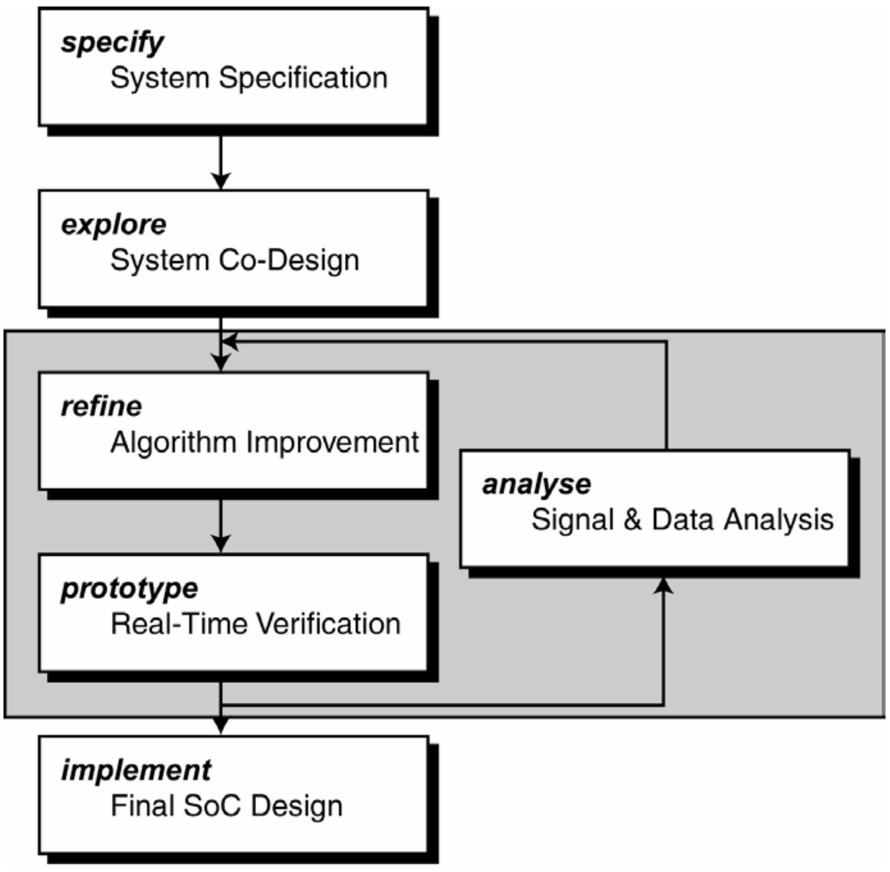
GECKO Design Environment

- Design entry:
  - C-code software
  - manual RTL hardware
  - algorithms
- All three design entry elements will be converted to VHDL and thus can be implemented into a SoC



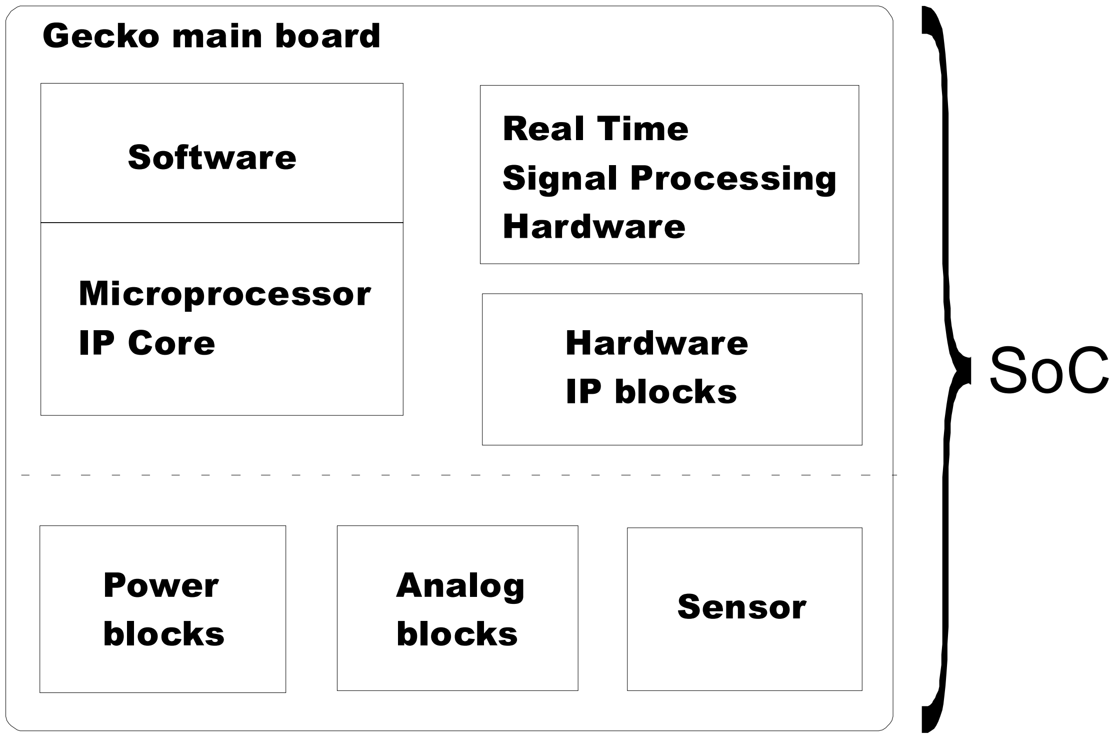
SoC Design Methodology

- The specify-explore-refine design flow is extended to a specify-explore-refine-prototype-analyze design flow for SoC designs with real-time constraints

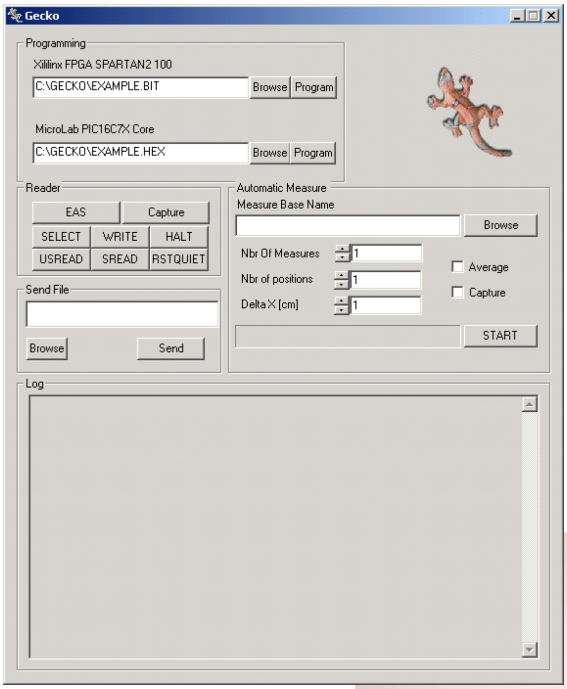


SoC with GECKO Environment

- An SoC design using the GECKO system supports the two chip approach
  - GECKO main board for digital part
  - application specific GECKO expansion board for analog, power, HF part



The GECKO system



Xilinx FPGA SPARTAN2 100: C:\GECKO\EXAMPLE.BIT

MicroLab PIC16C7X Core: C:\GECKO\EXAMPLE.HEX

Measure Base Name:

Nbr Of Measures: 1

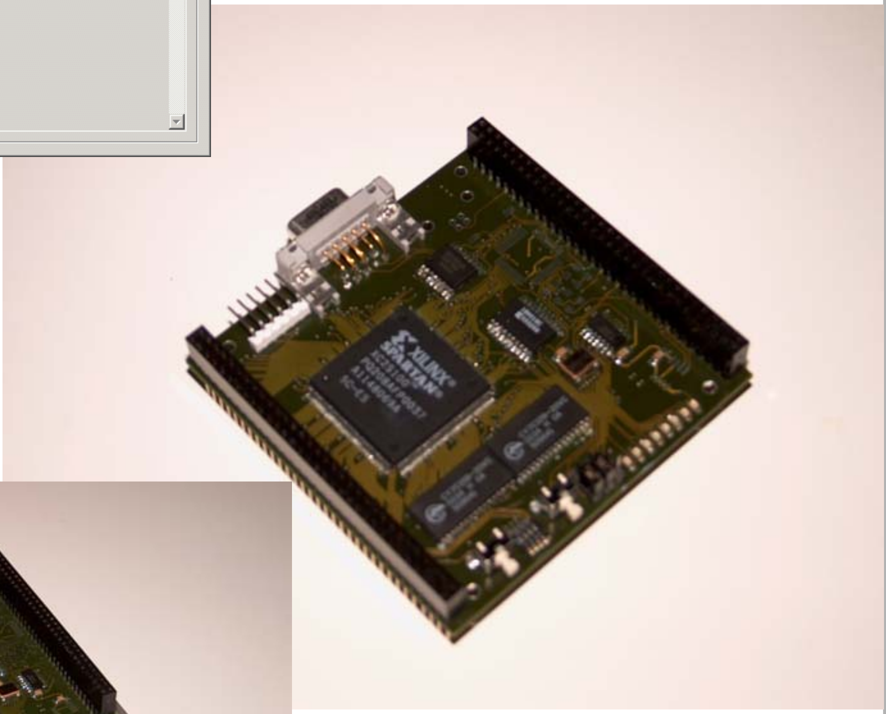
Nbr of positions: 1

Delta X [cm]: 1

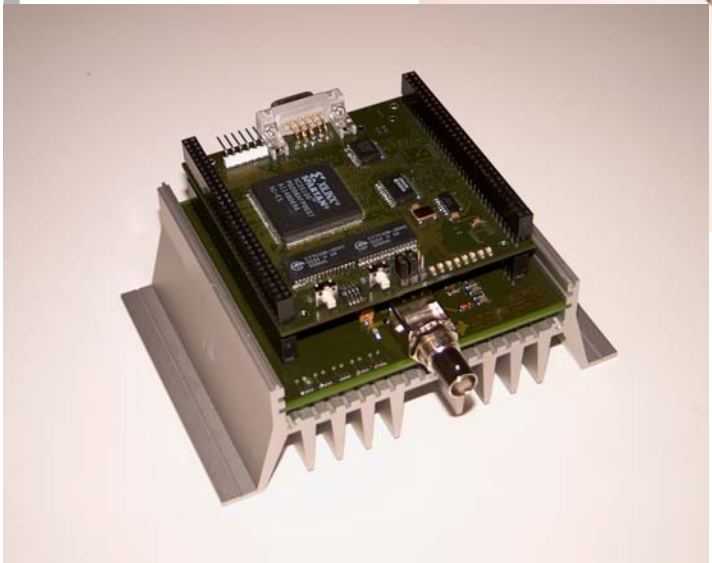
☐ Average

☐ Capture

GECKO Interface Driver



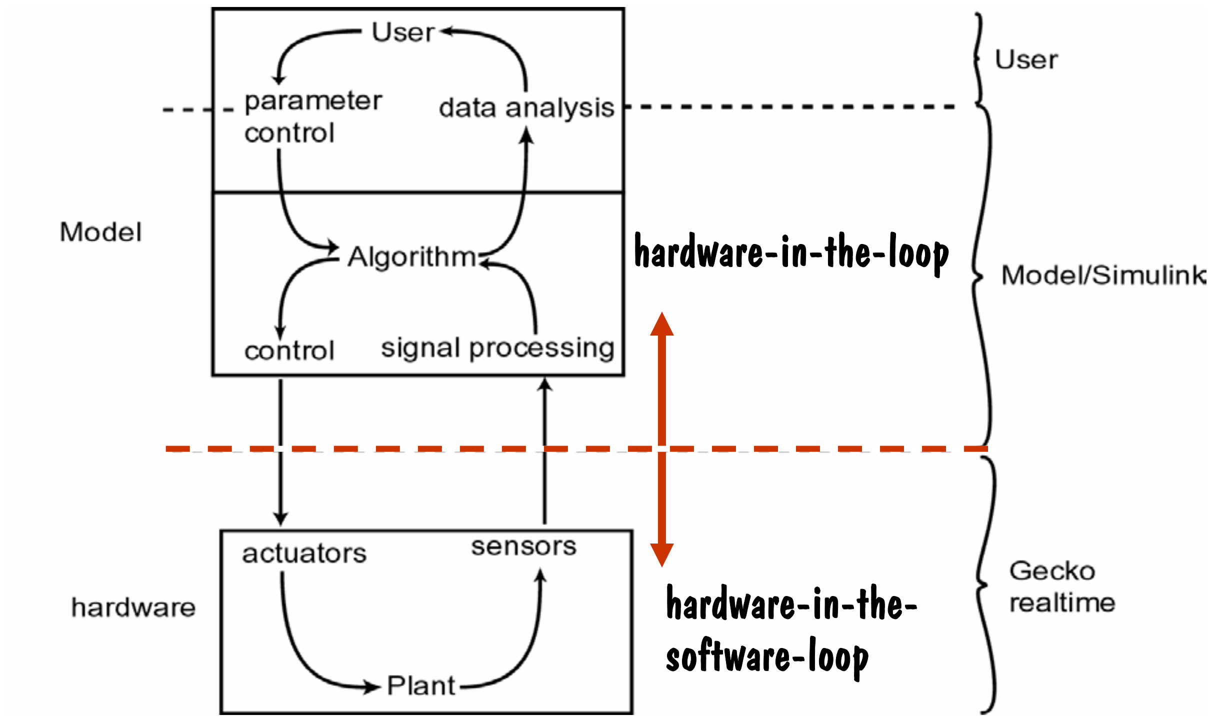
GECKO main board



GECKO main board n top if an application specific GECKO expansion board (RFID reader application, 2 W 13.56MHz RF power)

Hardware-in-the-Loop

- to iteratively improve a design fast prototyping and data analysis steps are necessary
- difficult to model plants are preferably not be modeled and directly included in the simulation loop
- variable cut between simulation and hardware
- respect real-time constraints



Homework: MyProject

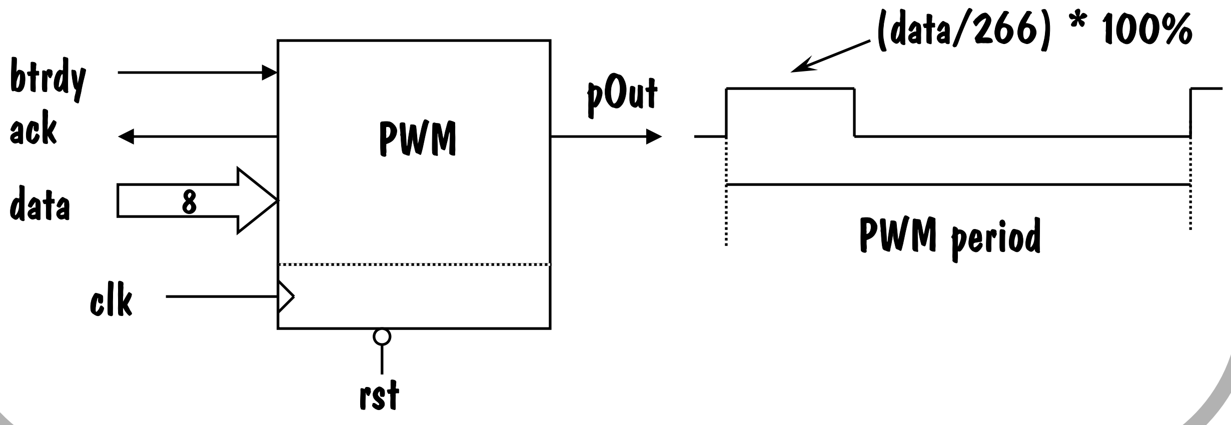
- define your own project
- plan the development and use the presented design methodology
- prepare the presentation of your project, be sure you do have all the necessary documentation for the discussed design steps

- MyProject 2004: speed controlled dc motor
  - Matlab/Simulink with speed controller
  - GECKO main board with dc-motor electronics
  - use hardware-in-the-simulation-loop
- Implementation constraints:
  - microprocessor with C code for „administrative“ tasks
  - pulse wide modulation for driving dc motor (hardware)
  - A/B signal encoder for speed sensing (hardware)
  - driving circuitry (expansion board) as simple as possible
- Technical data:
  - dc motor has 6000 turns/minute at 5V
  - speed sensor has 12 pulses per turn

Exercises: SoC #1

- **CAD Ex55x: PWM Project (difficulty: easy; time: medium):** Design of a pulse width modulator (PWM) controlling a DC-motor. The PWM shall have an microprocessor interface. The VHDL design is simulated, compiled and implemented into an FPGA and is supposed to drive small dc motor.

- **CAD Ex550: (difficulty: easy):** Design the VHDL code of the PWM element. The btrdy and ack signals are handshake signals for communication with the microprocessor data bus. A value 0 on the 8-bit data bus will switch off the dc motor (pOut='1'), a non-zero value will generate a PWM signal with an on-time of (data/256)*100% of a period. Analyze the VHDL syntax with gvan.



Exercises: SoC #2

- **CAD Ex551: (difficulty: easy):** Design a test-bench for the PWM. Simulate your VHDL code with the Synopsis VSS simulator and use your test-bench to verify its correct behavior.

  Result: see exercise Ex451 on the MicroLab web

- **CAD Ex552 (difficulty: easy):** Synthesize the PWM VHDL code into a gate level schematic for a Xilinx FPGA target technology. Connect your VHDL signals to the correct FPGA pins. Perform the place&route of the logic elements.

  Result: see exercise Ex452 on the MicroLab web

- **CAD Ex553 (difficulty: easy):** Download your PWM circuit into an FPGA and and applying different PWM values to your circuit by the GECKO User Interface tool. Use an oscilloscope to verify its correct output behavior. This exercise has to be done in MicroLab, using the GECKO system.

  Result: see exercise Ex453 on the MicroLab web