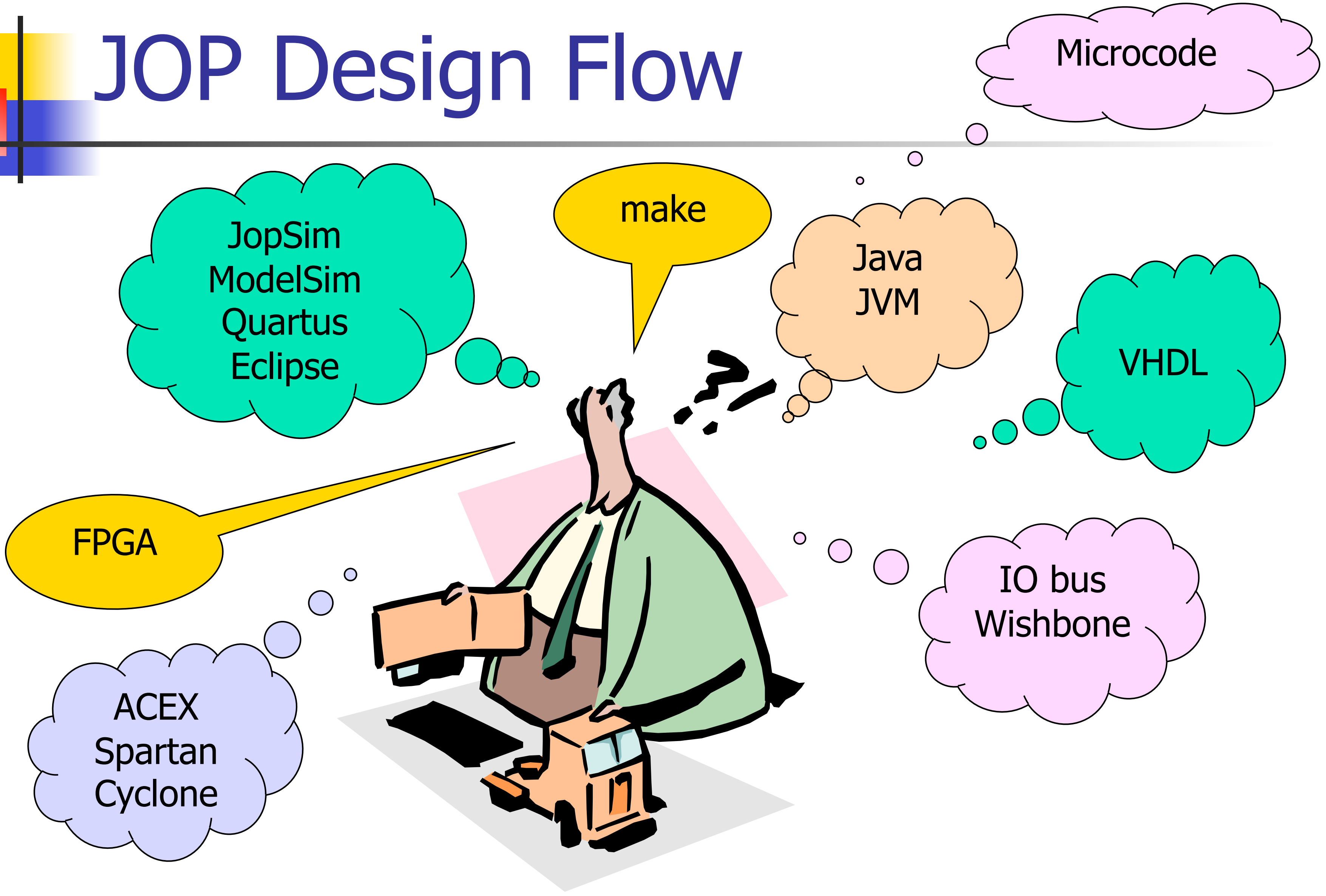
JOP Design Flow



The Project

- 4/5 programming languages
  - VHDL, Java, Microcode, C, (Verilog)
- 2648 files
- 498k LoC
  - Java: 350k
  - VHDL: 49k
  - Microcode (.asm): 4k
  - C: 4k
- 5 FPGA types, 9 target boards
- Docu: 19k LoC

Don‘t Panic

- This complexity is not unusual
  - Linux kernel: 6M LOC!
- There is a master `Makefile`
  - A single target can be built
  - Calls batch files
- Help
  - *Some* documentation is available (pdf and web)
  - e-mail to Schoeberl
  - Yahoo! Groups : java-processor

Directories

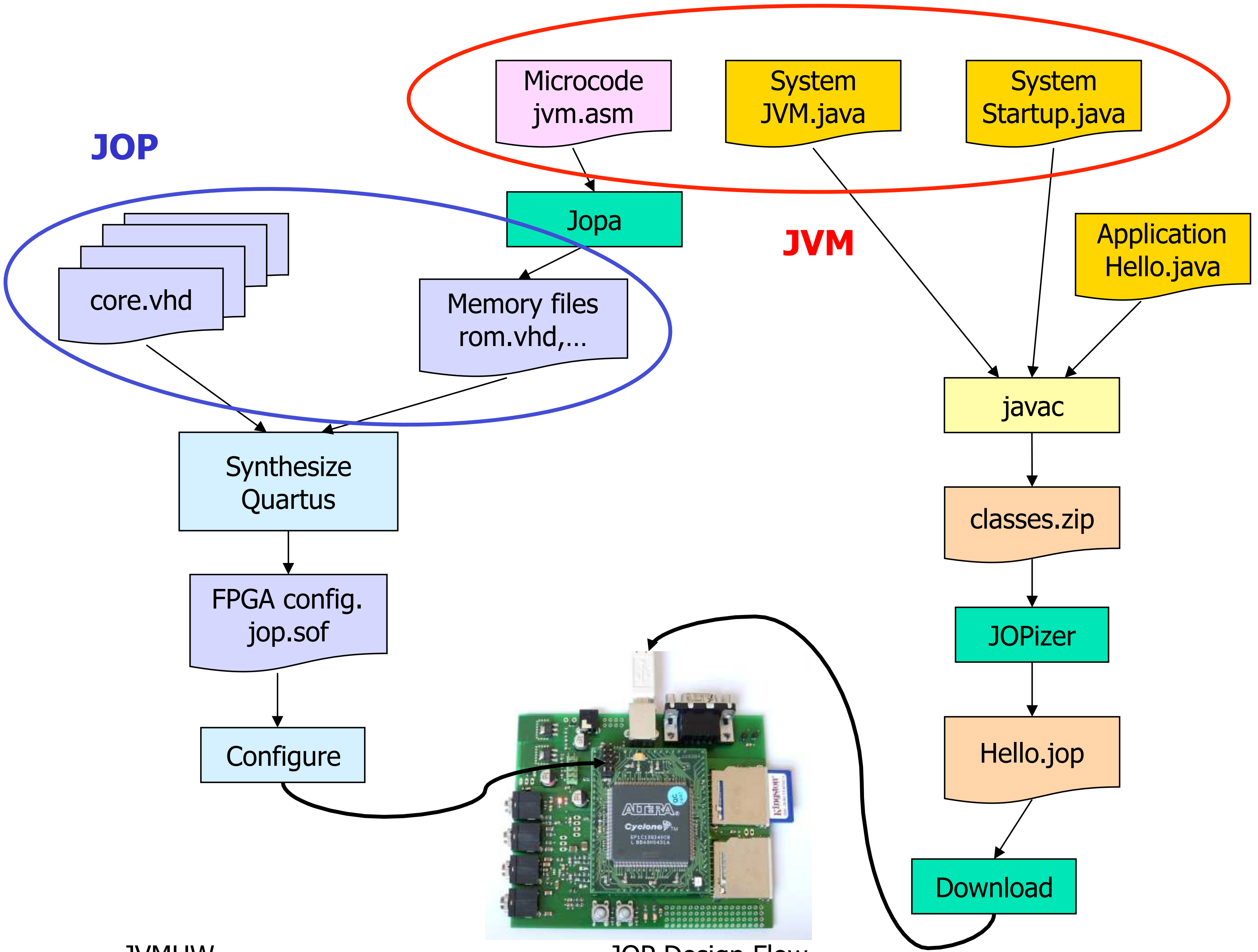
- vhdl
  - The hardware description of JOP
- asm
  - The JVM in microcode
- java
  - System sources (JVM, JDK)
  - Target applications
  - Tools

Directories cont.

- quartus
  - Project files for Altera FPGA boards
  - Each board and variation in its own directory
- xilinx
  - Project files for Xilinx FPGA boards

File Types

- Source
  - .asm, .vhd, .java
- Generated
  - JVM assembly: .vhd, .mif, .dat
  - Quartus: .sof, …
  - JOPizer: .jop
- Configuration
  - Project: Makefile, .bat
  - Quartus: .qsf, .cdf


JOP
Microcode
jvm.asm
System
JVM.java
System
Startup.java
Jopa
JVM
Application
Hello.java
core.vhd
Memory files
rom.vhd,…
javac
Synthesize
Quartus
classes.zip
FPGA config.
jop.sof
JOPizer
Configure
Hello.jop
Download

JOP Startup

- FPGA configuration
  - ByteBlaster download cable
  - USB
  - Flash on power up
    - Watchdog -> PLD configures FPGA
- Java application
  - Serial line
  - USB
  - Flash

Startup Configuration

- FPGA configuration
  - PLD (MAX7064)
  - ByteBlaster: cyc_conf_init.pof
  - Flash: cyc_conf.pof
- Java application
  - JVM (jvm.asm) on startup
    - Loads the application (.jop)
    - Defines download type
    - Constants: FLASH, USB, SIMULATION

Targets

- Top level defines FPGA type
  - jopcyc.vhd
  - jopcyc12.vhd
  - jopacx.vhd
  - …
- IO top level defines board type
  - scio_min.vhd
  - scio_baseio.vhd
  - scio_dspio.vhd
  - …

JVM + Library

- JVM
  - Microcode (jvm.asm)
  - Java (JVM.java, Startup.java, GC.java)
- Library
  - JOP specific: util, ejip, joprt
  - JDK: System, String,…

Native Functions

- Bridge between Java and the HW
  - Memory, IO access
  - Register, stack cache
- Special bytecode
- Implemented in microcode
- Translated by JOPizer
- Define in:
  - jvm.asm
  - com.jopdesign.sys.Native.java
  - com.jopdesign.tools.JopInstr.java

Simulation

- VHDL with ModelSim
  - HW related changes
  - Testbench reads the memory content
- High level with JopSim
  - System debugging (e.g. GC)
  - Reads .jop files
  - A JVM in Java
- Board simulation

Summary

- Modules
  - JOP – VHDL files
  - JVM – Microcode + Java
  - Application – Java
- Build
  - Jopa, Quartus -> FPGA configuration file (.sof)
  - javac, JOPizer -> Java application file (.jop)
  - Makefile + Batchfiles

More Information

- Chapter 2 in the handbook
- An Introduction to the Design Flow for JOP

Next Lecture

- Topic: The Java Virtual Machine
- We 9.6.2010, 10:00, Room 030
- Recommended preparation:
  - JOP Thesis: p 7-16, p 55-64, JOPThesis
  - Handbook: p 35-46
  - Look into a very simple JVM e.g.:
    - JopSim.java (in java/tools/src/com/jop…)
    - http://www.jopdesign.com/download.jsp
  - Disassemble simple Java files with `javap`

Lab

- Starts today in Databar 229
- Do preparation examples
  - Build JOP and change simple things
- Start to think about the project
- See you at 13:00!