SOFTWARE DRIVES HARDWARE, LESSONS LEARNED AND FUTURE DIRECTIONS

Rob Oshana
Vice President, Software Engineering, R&D
NXP Semiconductors
Microcontroller and Microprocessors

RISC-V

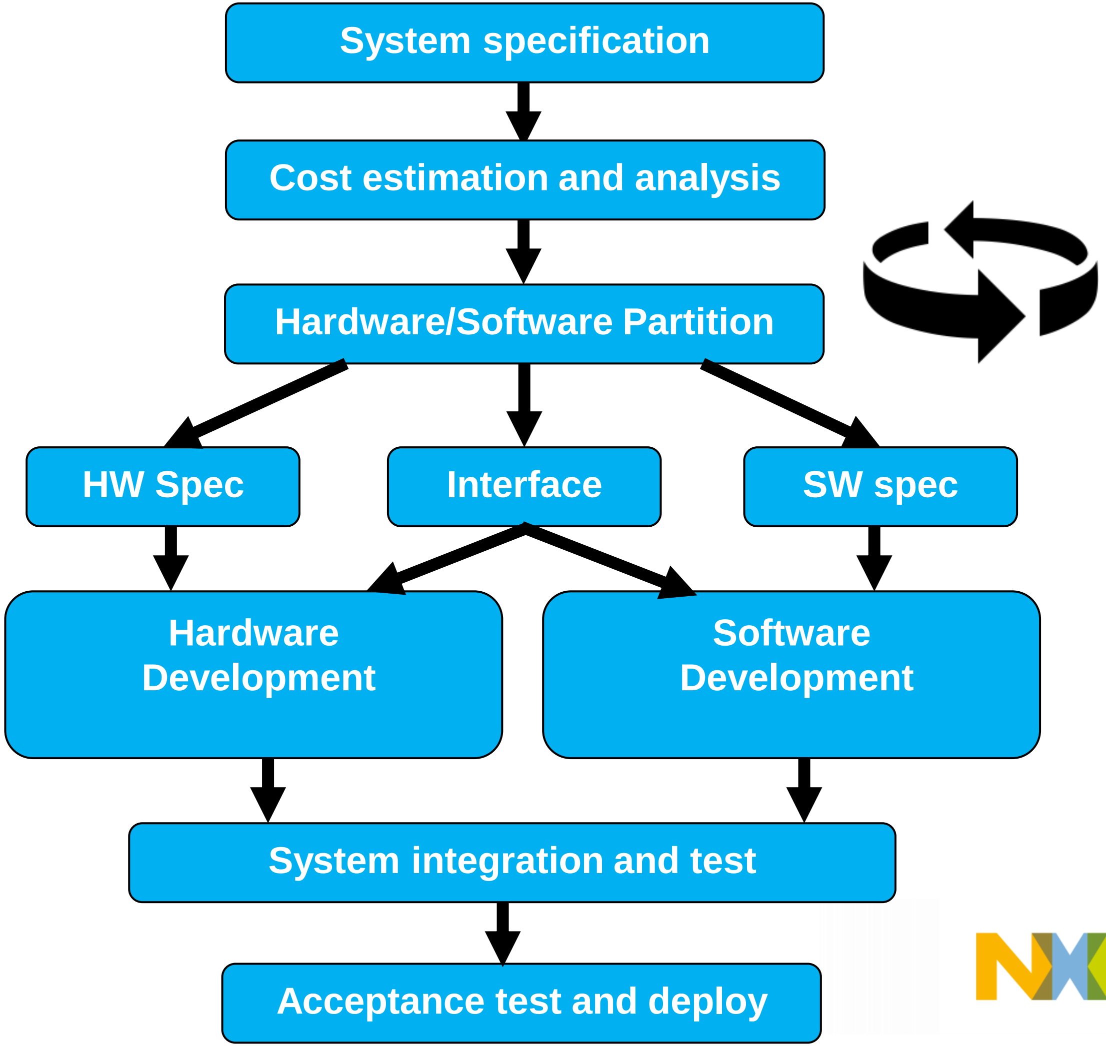
Key messages

- Software engineers can innovate earlier/often and drive more specific core requirements for hardware design team

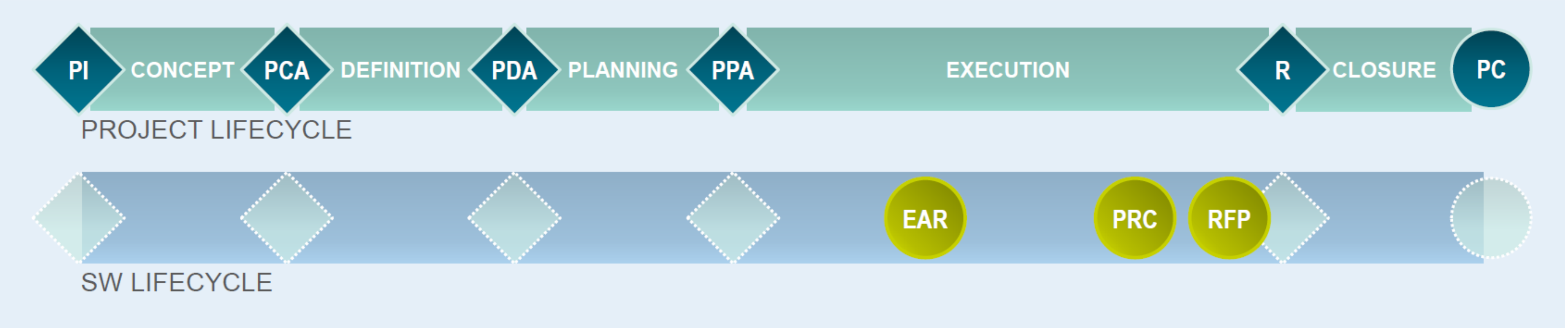
- Similar OSS model evolving for RISC-V

- Industrial “stress test” of these trends seems to validate the approach for embedded system development

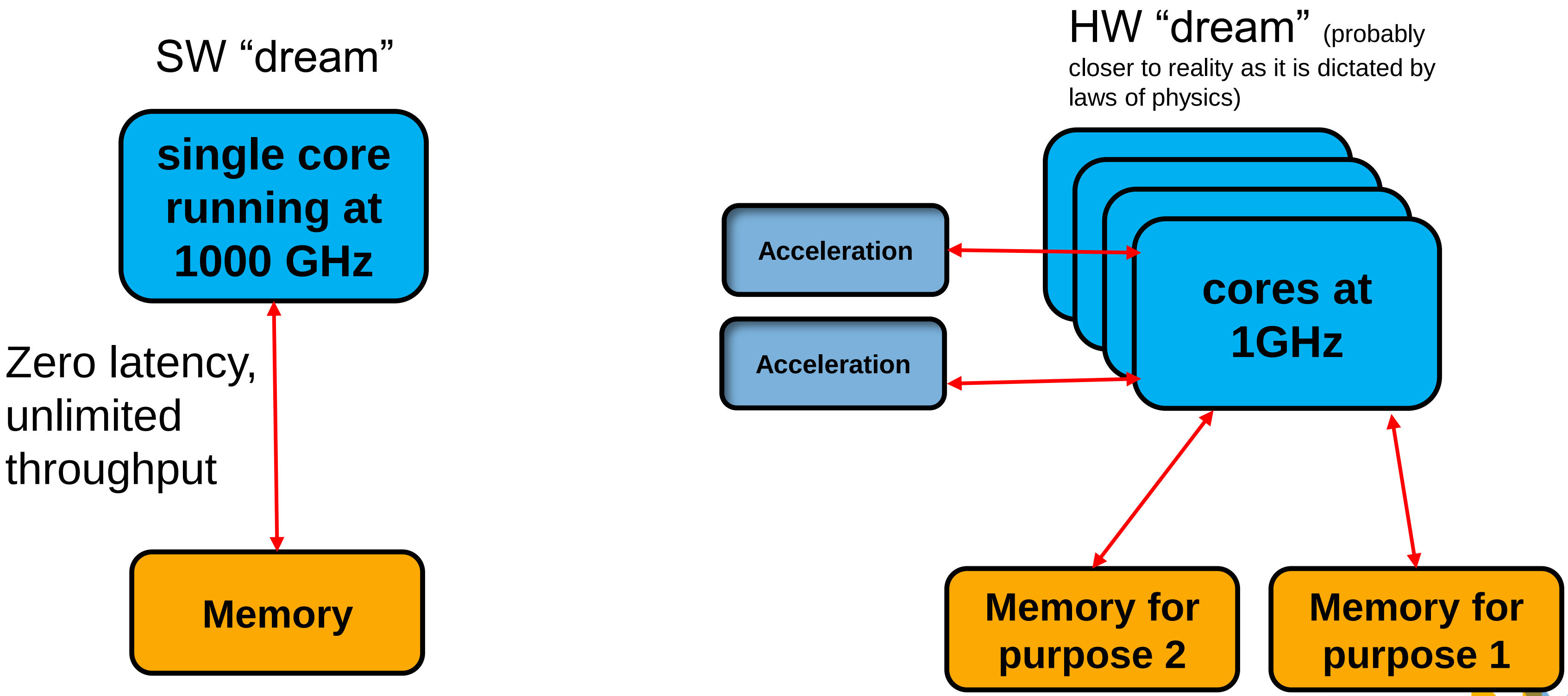
Traditional HW/SW co-design



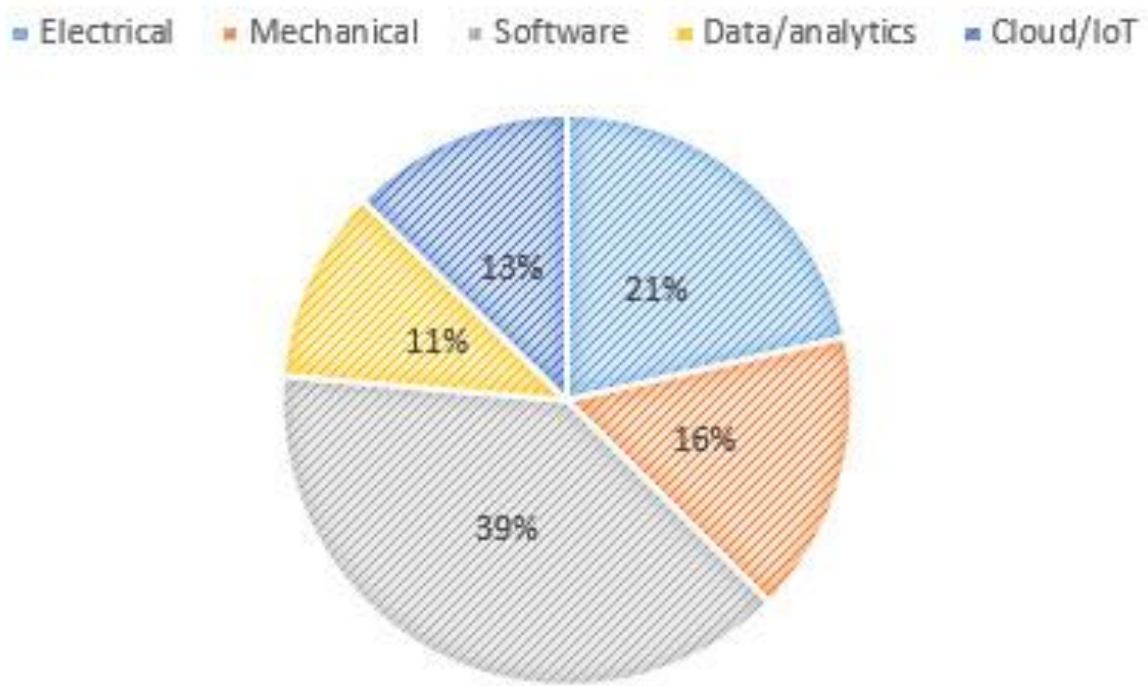
An Embedded Product Build Process (Industrial)



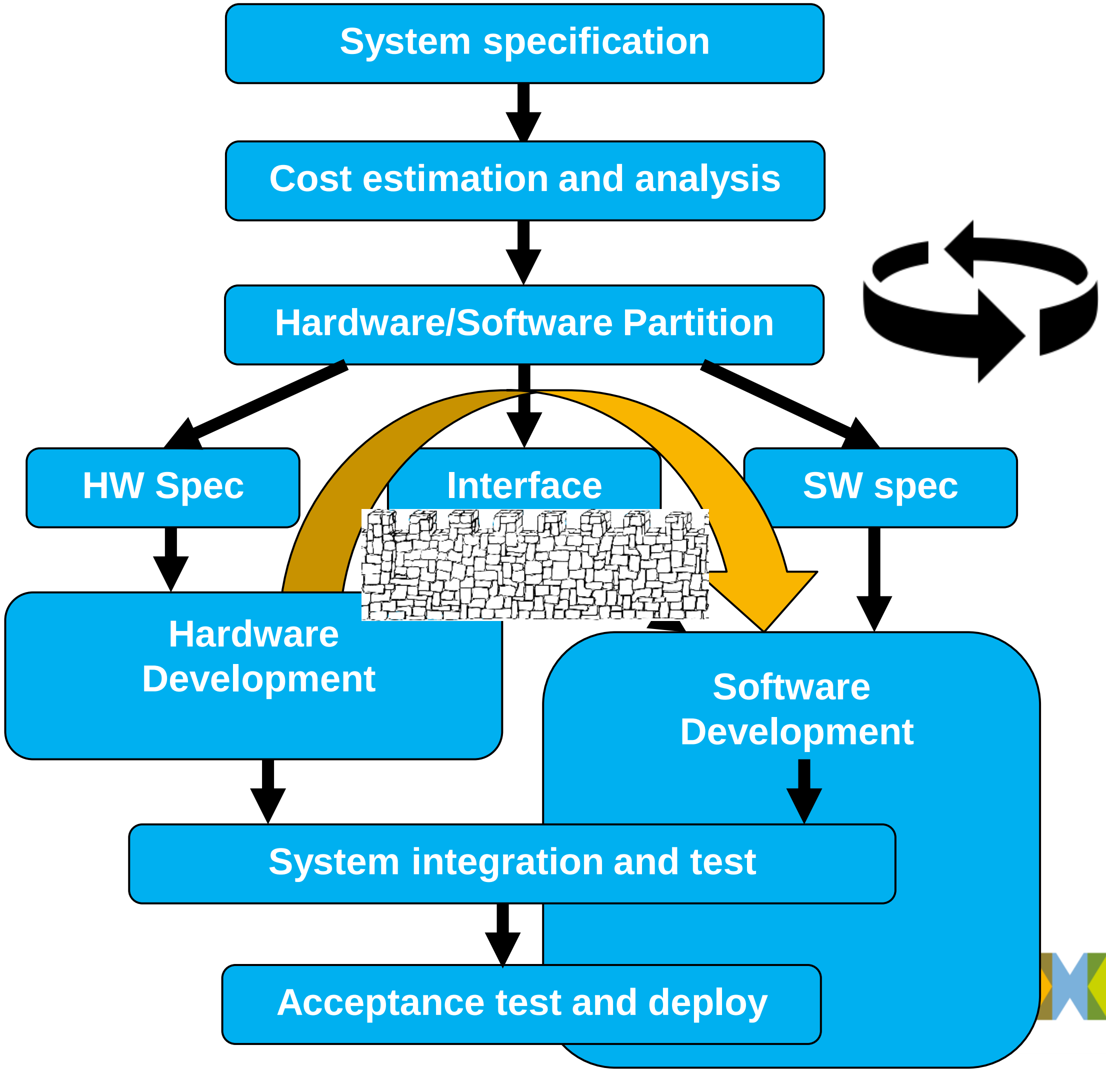
HW engineers are from Mars SW engineers are from Venus



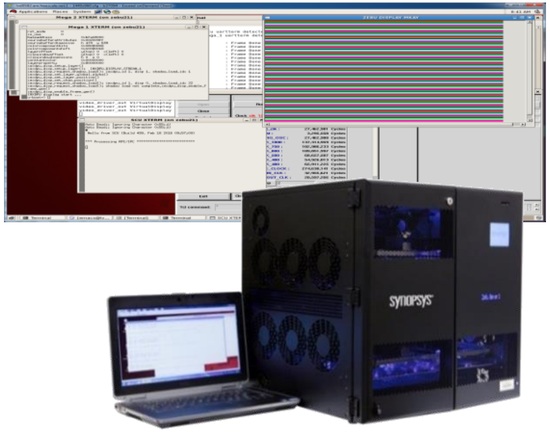
Software consumes more time that any other aspect of an embedded project (VDC Research)

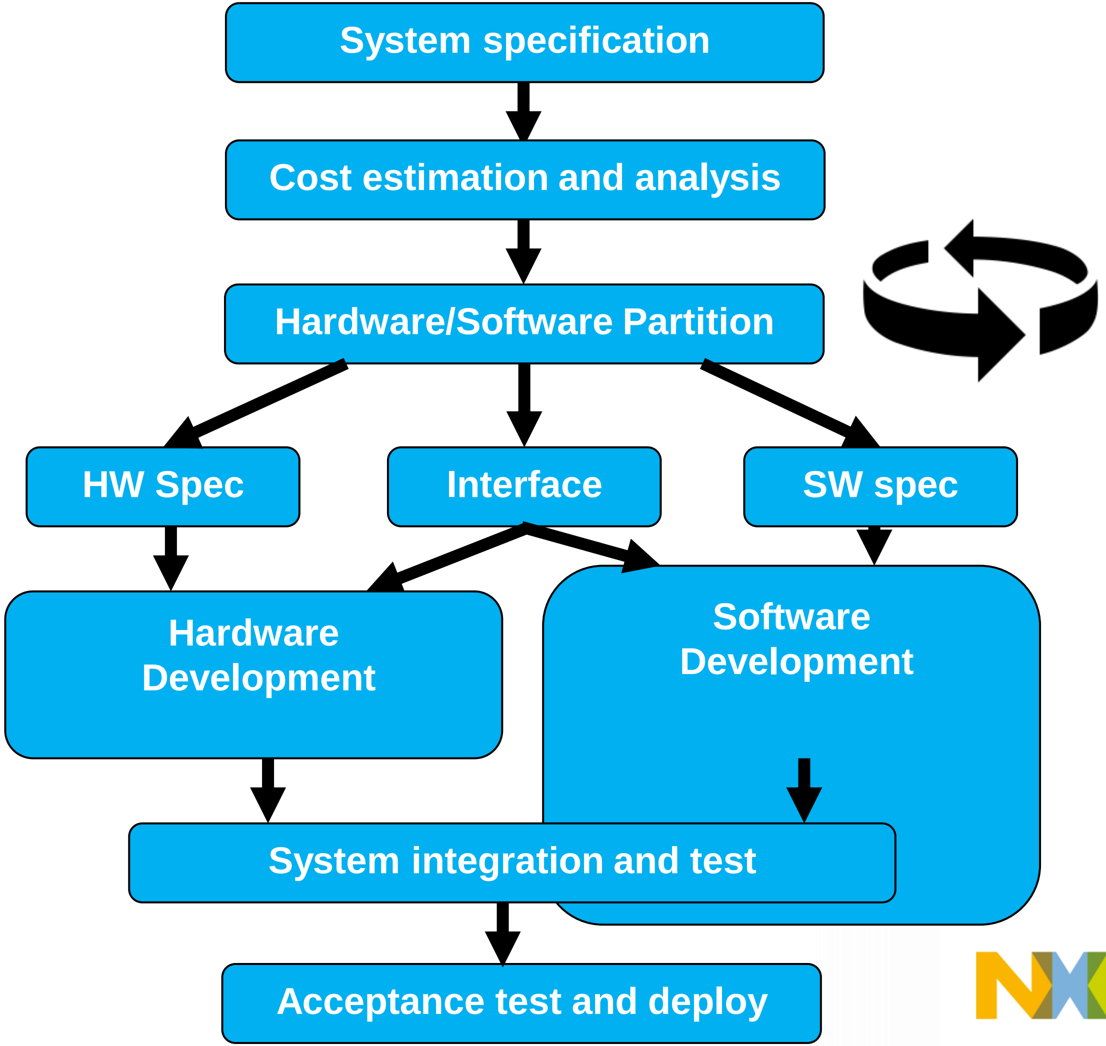


Reality of HW/SW co-design



“Shift Left” Software Development for Embedded Systems (circa 2010)

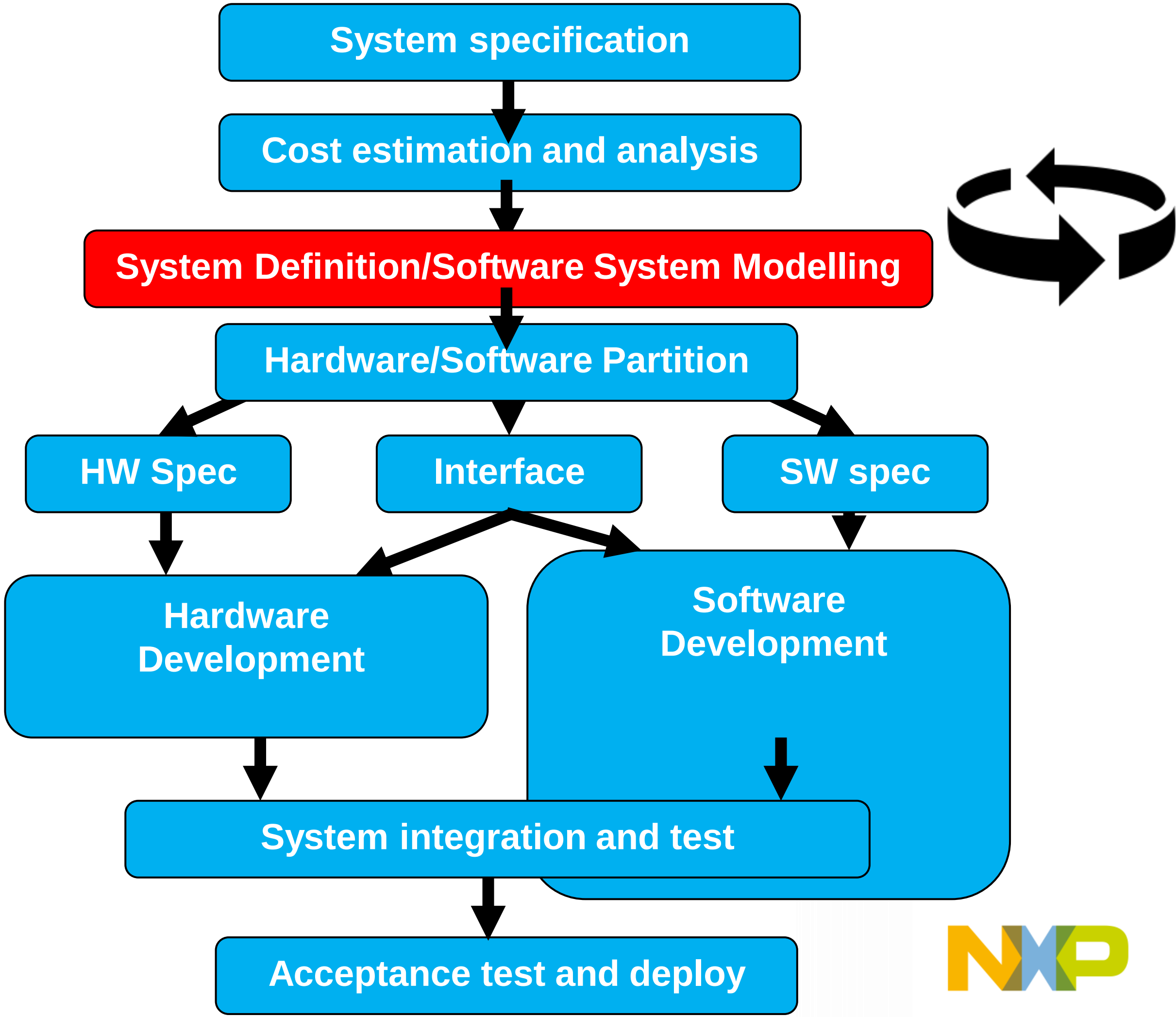




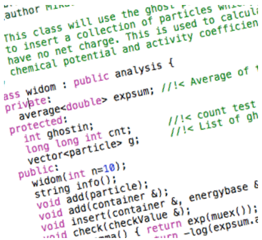
“Software Drives Hardware” Approach to Embedded Systems w/ RISC-V

Programming model; languages and libraries that create an abstract view of the machine

- Control
- Data
- Synchronization



New deliverables by software systems engineers

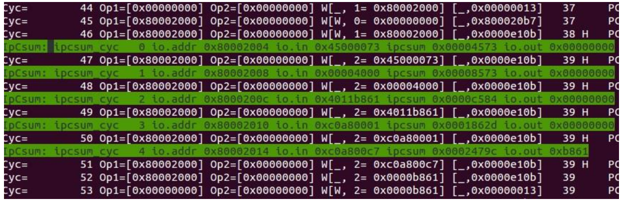


System modeling
(by SW Eng)

Models (C/C++, MATLAB, etc)
Performance requirements

⬇

System definition



⬇

- **Intrinsic libraries in standard format**
- **New instructions**
- **Programming model details**
- **Chisel model**

Embedded Systems and Software – Past, Present, and Future

- 1980; a few drivers after the HW is developed
- 1990; commercial RTOS to handle increased SW complexity
- 2000; early SW development on simulation and models
- 2010; “Shift Left” SW development for large scale software systems
- 2020; “Software Driven Hardware” to support SW programming model

OSS in Embedded Applications is Mainstream

- Linux in embedded projects shows an annual growth rate of about 50 percent

- Among the projects using Linux, about 80 percent use free public Linux distributions (500+)

- When the benefits of open source software are utilized, this oftentimes results in a more robust and flexible product

Free Software Foundation “Freedoms”, can these apply to RISC-V?

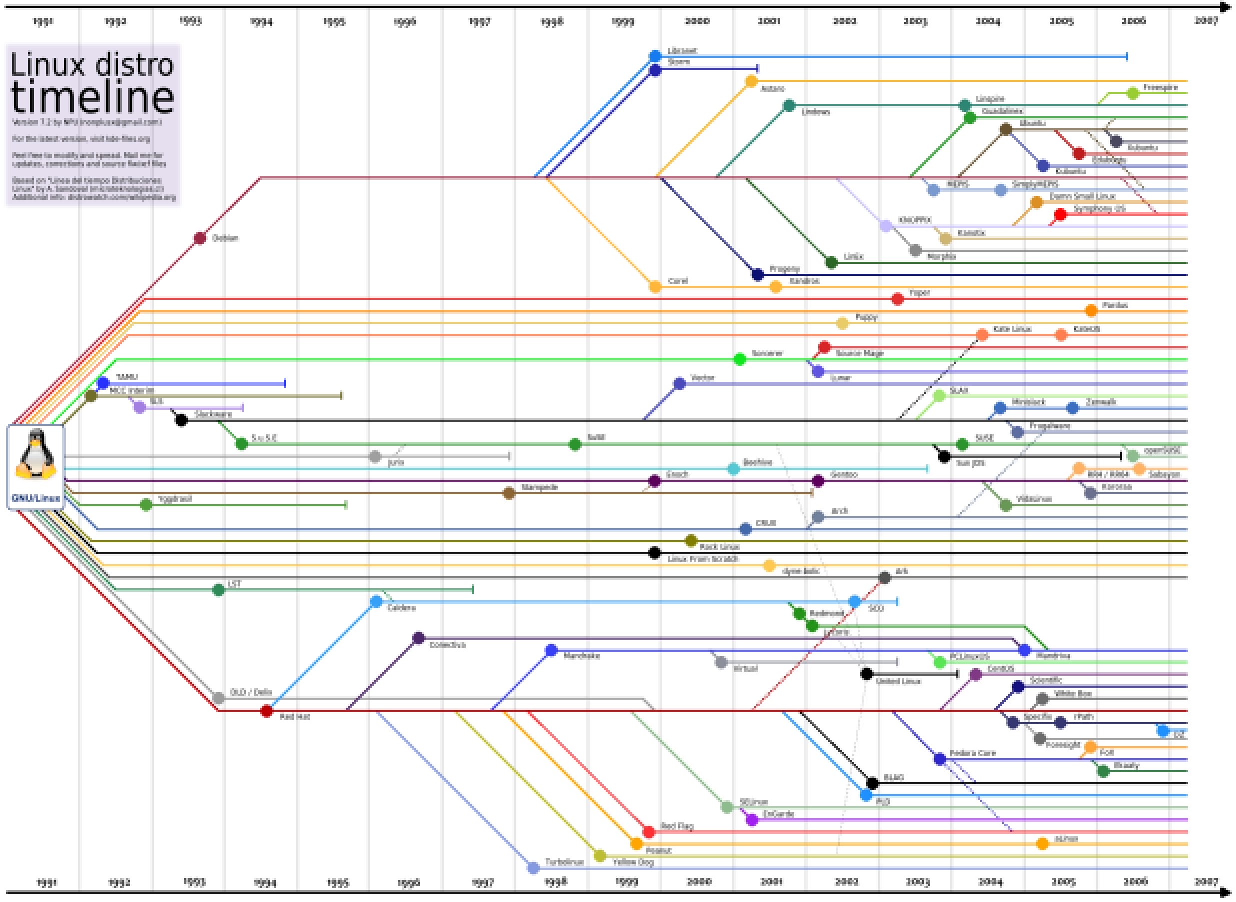
- “Free software” is a matter of liberty, not price (“free speech” not “free beer”)

Four Freedoms;

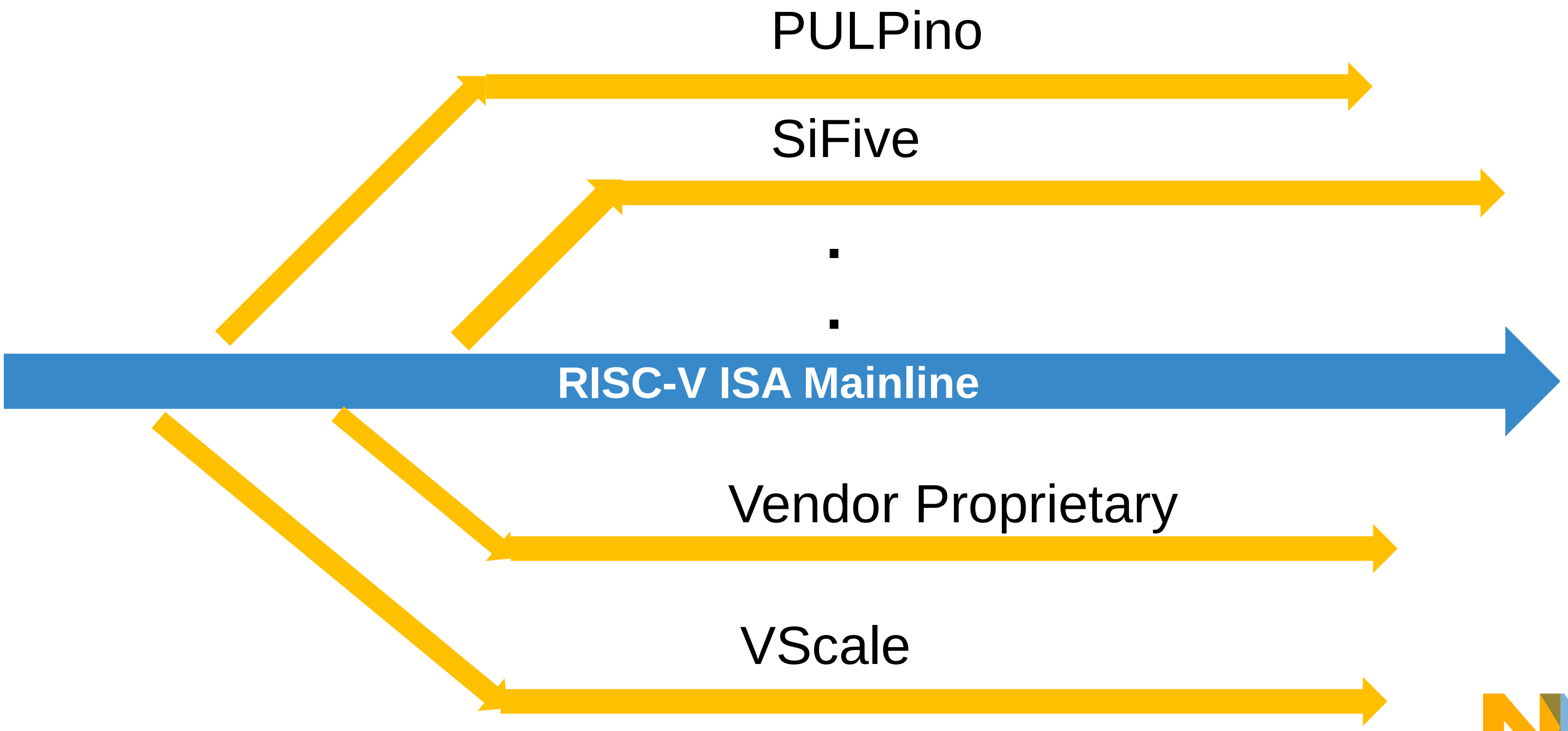
1. The freedom to run the software, for any purpose
2. The freedom to study how the software works, and change it to make it do what you wish. Access to the source code is a precondition
3. The freedom to redistribute copies so you can help your neighbor
4. The freedom to distribute copies of your modified versions to others

- By doing this you can give the whole community a chance to benefit from your changes.

Linux OSS Mainline and Contribution Model



Key Challenge: establishing a robust open community for RISC-V



Areas where NXP is innovating with RISC-V

- For APs + MCUs, we typically use a subsystem-based SoC design methodology
  - Often, the “black-box” subsystem designs include:
    - A “minion core” as an independent processing element
    - No user programmability, in-house developed firmware only
- Initial RISC-V uses target these types of black-box subsystem minion cores
  - As a small, processing element “controller core”
    - Focused on efficient core designs + ISA enhancements for application-specific functionality
      - Examples: bit manipulation operations, basic crypto operations for symmetric key & hash algorithms
      - Programmable SoC Miscellaneous Function Controller, Security Subsystem
- Other innovation targets include a Multi RISC-V Core MCU SoC
  - Reusing open-source MCU cores + existing dual-core hardware infrastructure

Industry effort to incorporate RISC-V “open source”

Design tasks	Design Effort	Verification Effort
Fix bugs	🟡	🔴
Architecture alignment	🔴	🟢
Add new features	🟡	🔴

Software Driving Hardware and FORTL (Free and Open RTL)

- Embedded software engineers will take a bigger role in defining the SoC architecture
  - Programming model
  - System optimization
- Open source RISC-V implementations will allow more Software driven Hardware
  - e.g. community open source applications are less entropic than proprietary applications
- Communities must learn from lessons of other open source efforts
- Ecosystem is vital to success


NXP
THANK YOU!