

---

# EEL 4783: Hardware/Software Co-design with FPGAs

## Lecture 2: Hardware vs. Software

Prof. Mingjie Lin



# Hardware vs. Software

---

- Model  $\neq$  Actual implementation
- Hardware can be modeled as RTL programs
  - RTL defines network of logic gates
- Software can be modeled as C programs
  - C program models binary instructions
- Hardware/Software Codesign
  - We work with models that partly written as C programs and partly as RTL programs
  - This class uses Verilog/VHDL and C programs

# Defining H/W Codesign

---

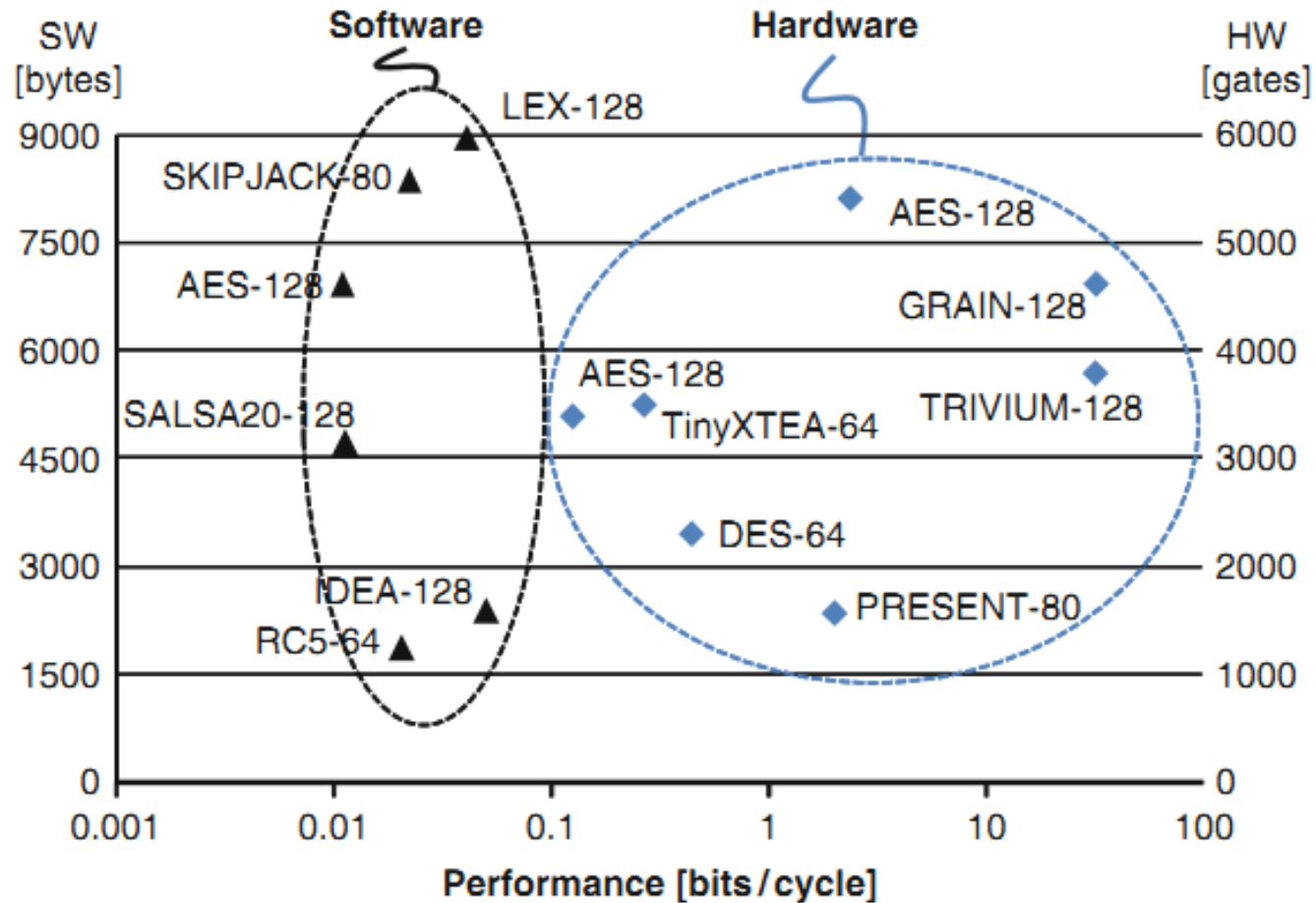
- The design of cooperating hardware components and software components in a single design effort
  - Hardware examples
    - Field Programmable gate Array (FPGA)
    - Soft-core processor
    - Digital-Signal Processor (DSP)
    - Application-Specific Instruction-set Processor (ASIP)
    - CELL processor, used in the Playstation-3
  - The partitioning and design of an application in terms of fixed and flexible components
-

# Why not all software?

---

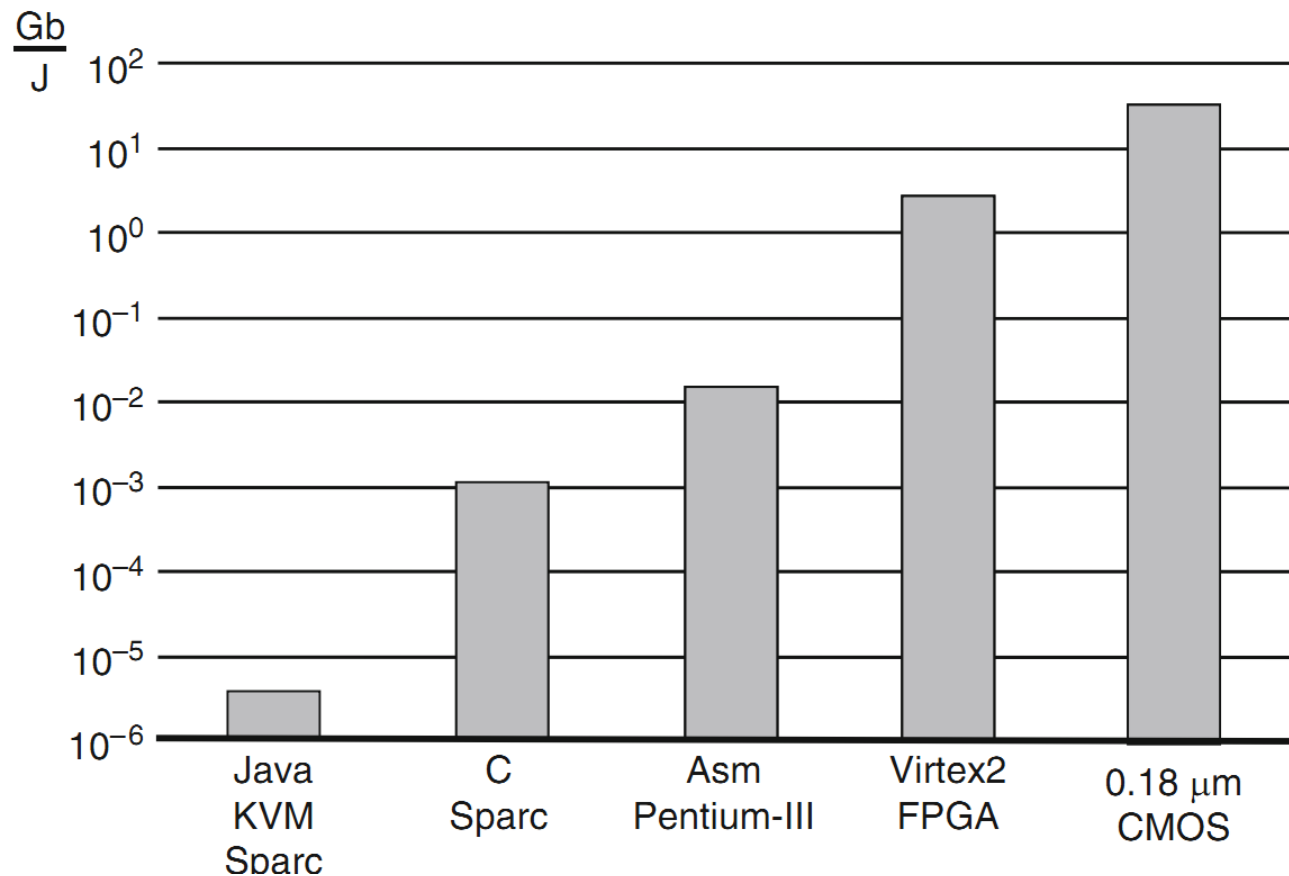
- Software is
  - Easy and flexible,
  - Software compilers are fast
  - Large amounts of source code available
  - All you need to start development is a nimble personal computer
  
- So why
  - go through the effort of designing a hardware architecture when there is already one available (namely, the RISC processor)?

# Relative Performance



# Energy Efficiency

- The amount of useful work done per unit of energy



# Driving Factors in H/S Codesign

---

- Performance + Energy Efficiency + Power Density + Design Complexity + Design Cost + Shrinking Design Schedule

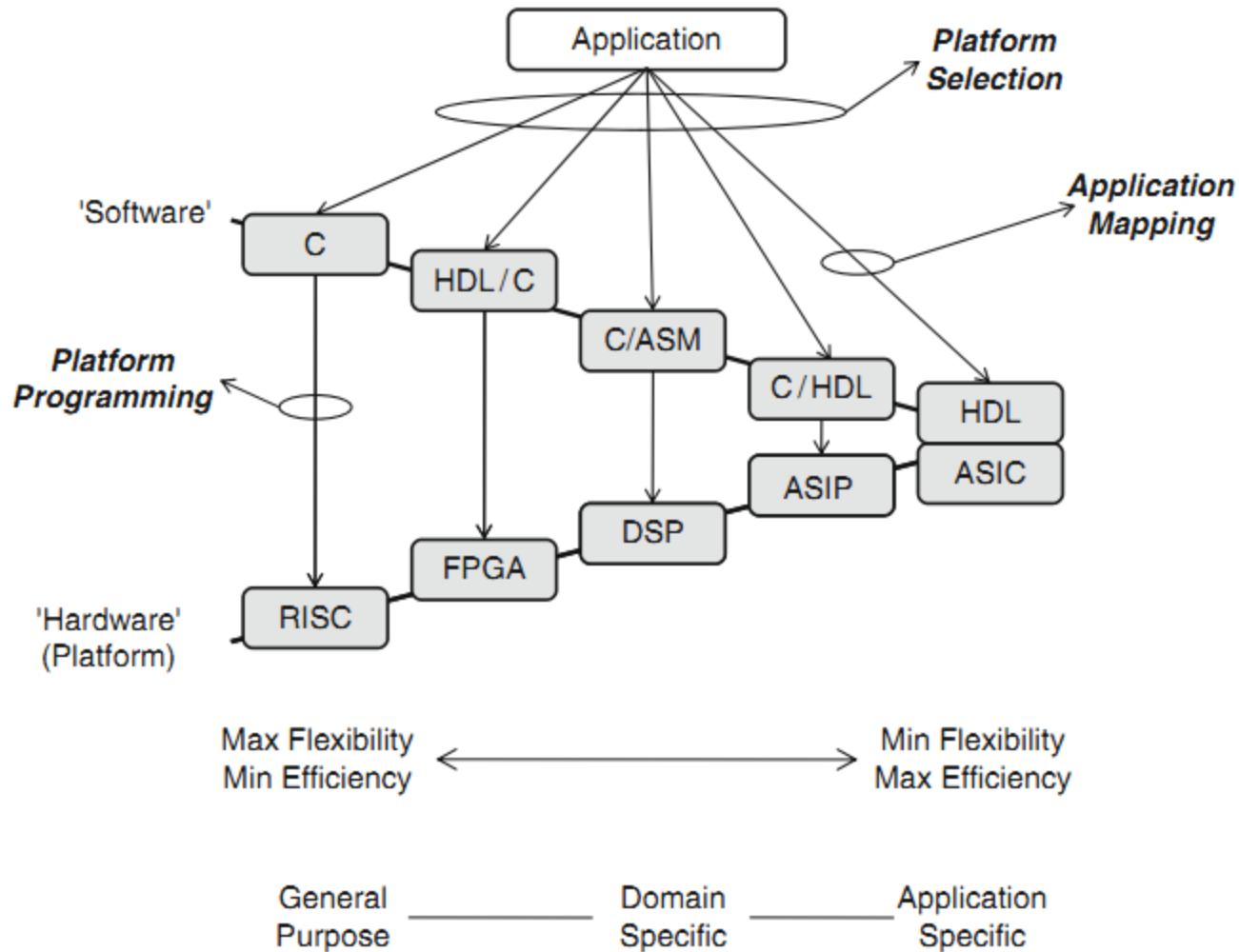
Improve Performance  
Improve Energy Efficiency  
Reduce Power Density

**Implement  
more in Hardware**

Manage Design Complexity  
Reduce Design Cost  
Stick to Design Schedule  
Handle Deep Submicron

**Implement  
more in Software**

# H/S Codesign Space





# Dualism of H/S Codesign

---

|                 | Hardware                    | Software                    |
|-----------------|-----------------------------|-----------------------------|
| Design paradigm | Decomposition in space      | Decomposition in time       |
| Resource cost   | Area (# of gates)           | Time (# of instructions)    |
| Constrained by  | Time (clock cycle period)   | Area (CPU instruction set)  |
| Flexibility     | Must be designed-in         | Implicit                    |
| Parallelism     | Implicit                    | Must be designed-in         |
| Modeling        | Model $\neq$ Implementation | Model $\sim$ Implementation |
| Reuse           | Uncommon                    | Common                      |

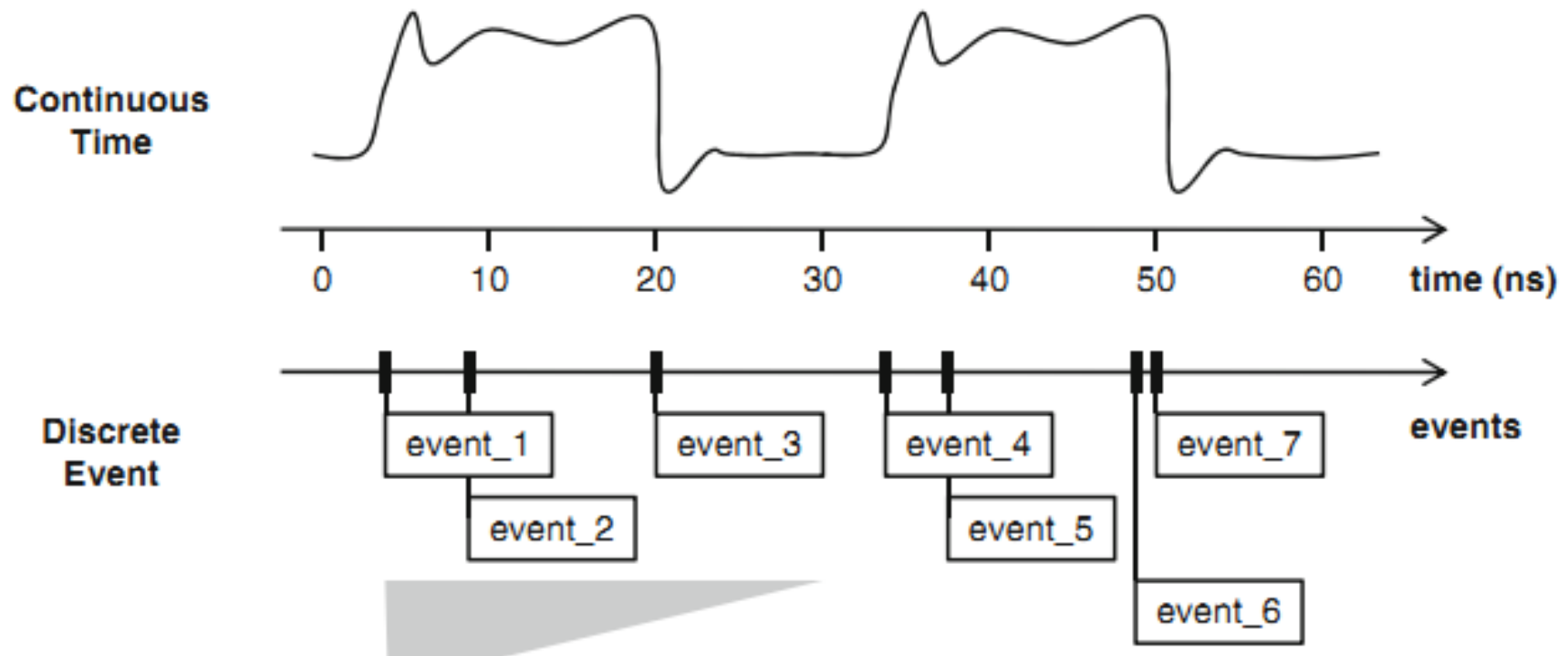
---

# Modeling Computation

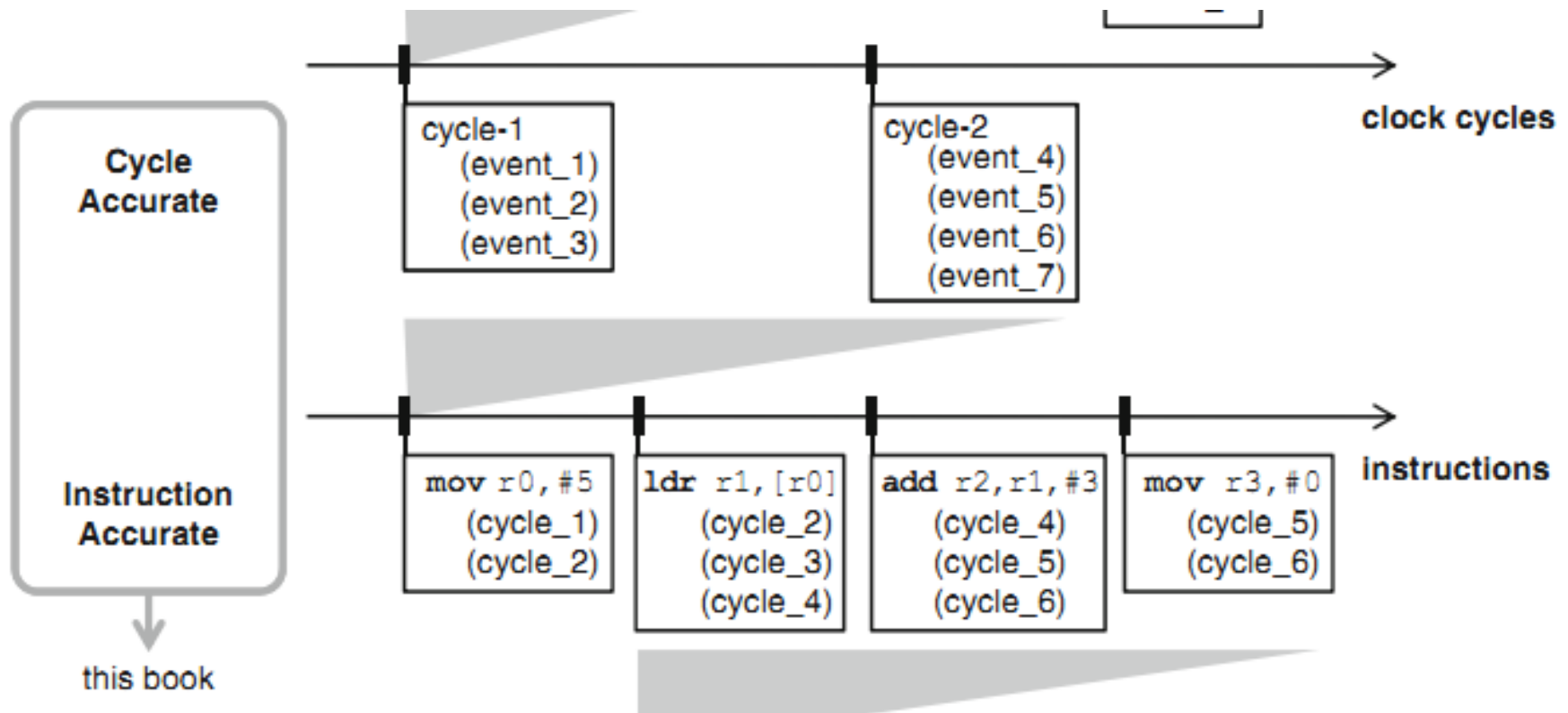
---

- Continuous Time
  - Lowest level
  - Operation -> continuous action
- Discrete-event
  - low level
  - Lump activity at discrete points in time called events
  - Removes differential equations and the complexity of continuous-time simulation
- Cycle-accurate
  - In single-clock synchronous hardware circuits, all interesting things happen at regularly spaced intervals, namely at the clock edge
  - Does not capture propagation delays or glitches

# Abstraction Levels



# Abstraction Levels



# Concurrency & Parallelism

---

- Concurrency: relates to an application model
- Parallelism: relates to the implementation of that model

# Lecture schedule

---

See Webpage:

[www.eecs.ucf.edu/~mingjie/EEL4783\\_2012](http://www.eecs.ucf.edu/~mingjie/EEL4783_2012)

# Final issues

---

- Come by my office hours (right after class)
- Any questions or concerns?