EN2911X: Reconfigurable Computing
Lecture 09: Design Flow: HW/SW Partitioning (1)

Prof. Sherief Reda
Division of Engineering, Brown University
http://scale.engin.brown.edu
Fall ’09

BROWN

Summary of current status

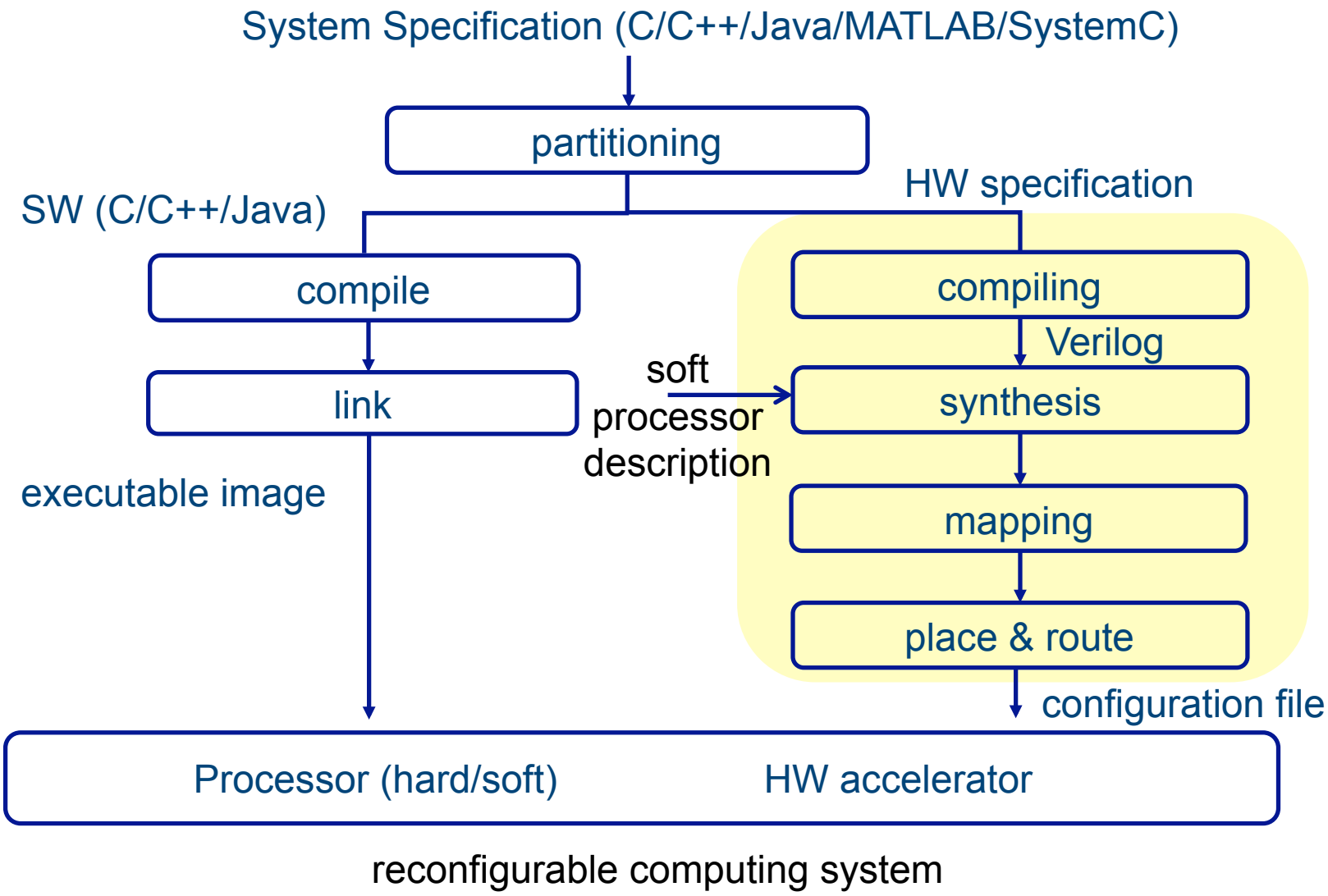
Past lectures:

- Understood the principles of the hardware part of reconfigurable computing: programmable logic technology.
- Learned how to program reconfigurable fabrics using hardware definition languages (Verilog).

Next lectures:

- Understand the principles of the software part (which we have partly used) of reconfigurable computing.
- Understand how to transform software systems into accelerated SW + HW systems.

Reconfigurable computing design flow



BROWN

System specification

Use High-Level Languages (HLLs) (C, C++, Java, MATLAB).

Advantages:

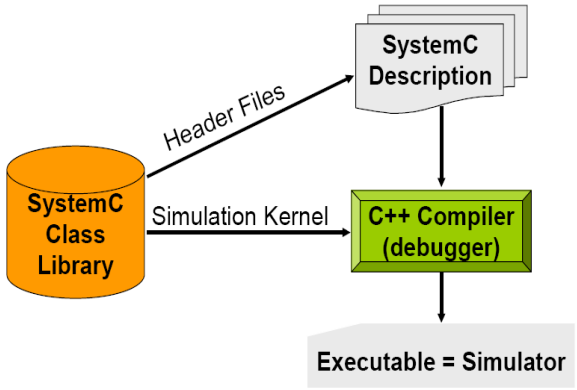
- Since systems consist of both SW and HW, then we can describe the entire system with the same specification
- Fast to code, debug and verify the system is working

Disadvantages:

- No concurrent support
- No notion of time (clock or delay)
- Different communication model than HW (uses signals)
- Missing data types (e.g., bit vectors, logic values)

- How can we overcome these disadvantages?

Using HLL for hardware/software specification



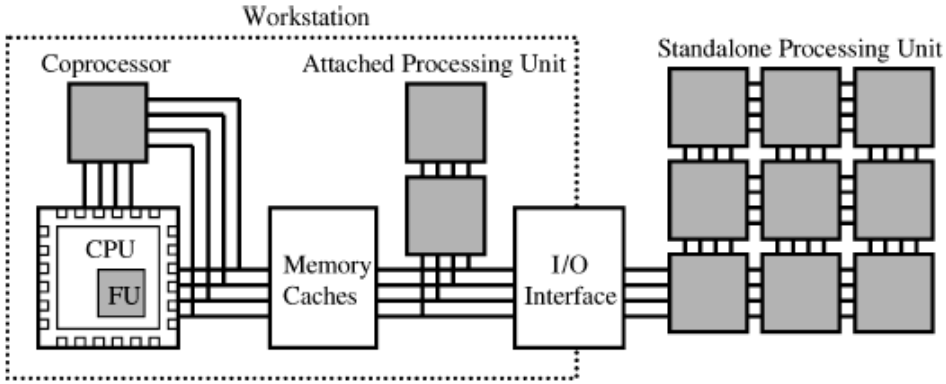
[from G. De Micheli]

- Augment the HLL (e.g. C++) with a new library, syntax that support additional hardware-like functionality (e.g. SystemC, ImpulseC)
  - Unified language across all stages of platform design
  - Fast simulation
  - There are already lots of tools for C++

  → we will come to this part later in details
- Enable compilers to optimize code and extract concurrency from sequential code to map into FPGAs

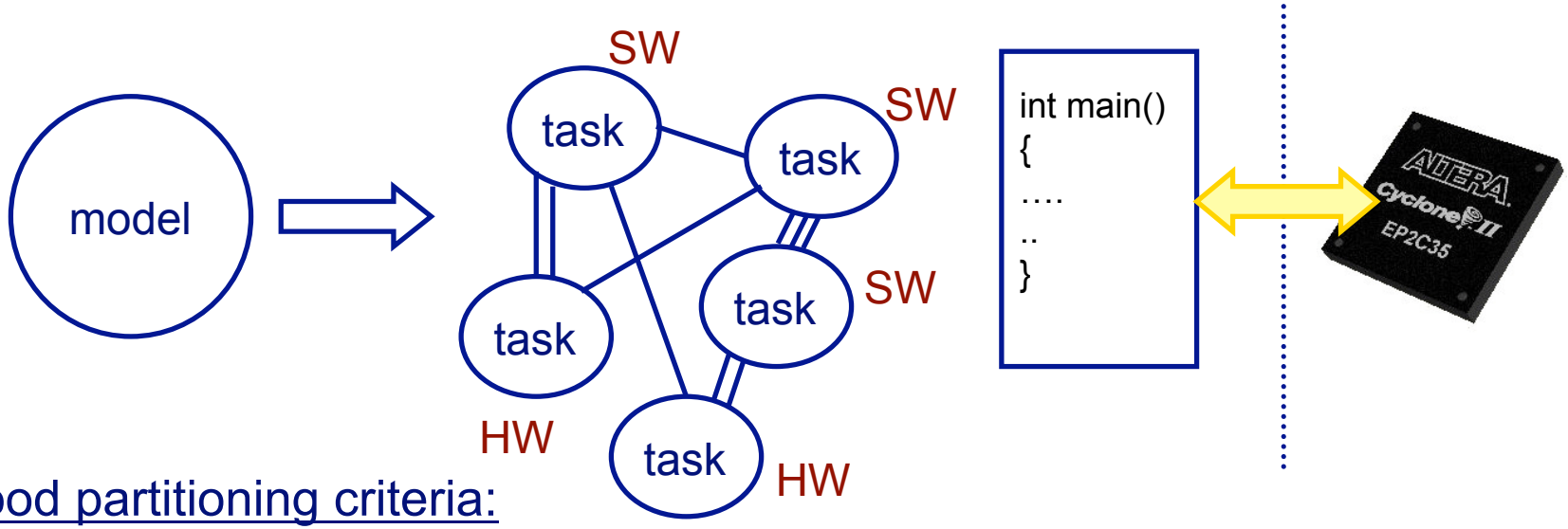
BROWN

Hardware-Software partitioning



- Given a system specification, decompose or partition the specification into tasks (functional objects) and label each task as HW or SW such that the system cost / performance is optimized and all the constraints on area resources / cost are satisfied.
- The exact performance depends on the computational model in hand
  - Given the same application, a system with an FPGA on a slow bus results in a design with different performance results than a system with a FPGA as a coprocessor.

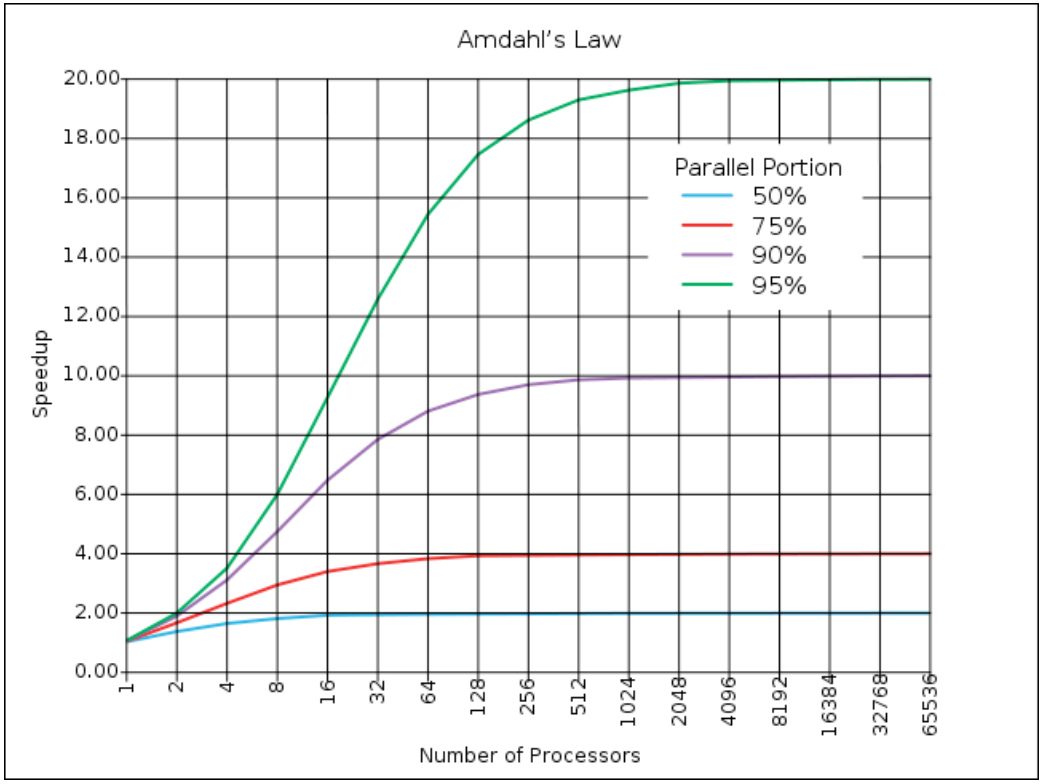
HW/SW partitioning



Good partitioning criteria:

1. Minimize total execution runtime or equivalently maximize speedup over just SW.
2. Minimize communication (traffic) between HW and SW and on the bus
3. Maximize concurrency (reduce stalling) where both the HW and SW run in parallel
4. Maximizes the utilization of the HW resources

Amdahl's law



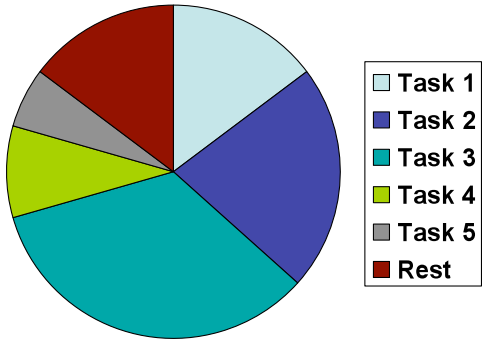
[from wikipedia]

- Application speedup is limited by the part that is not parallelized.
- For example, if 80% of the program can be transformed into parallelized HW then the program can be speed up at best by 5x

BROWN

Profiling is a key step in HW/SW partitioning

- Determining the candidate HW partitions by first profiling the specification tasks taking into account typical data sets



- Given a candidate SW/HW partition
  - Estimate HW implementation
  - Determine the system performance and speedup over software

- **How can we generate good candidate SW/HW partitions?**
  - ***PAPER REVIEW P04 for next Tuesday.***

Low-level partitioning from software binaries

- Rather than partition from the high-level description, it is possible to compile the program as SW and then partition the resultant executable binary into SW and HW parts.
  - Advantages:
    - No need to worry about which language is being used
    - Can be used to develop dynamic runtime partitioners and synthesizers
  - Main steps:
    - Decompilation of binary to recover high-level information
    - Partitioning and synthesis
    - Binary updating to account for the SW parts that migrated to HW