XILINX

Vitis Design Methodology

Introduction to Vitis

Goal

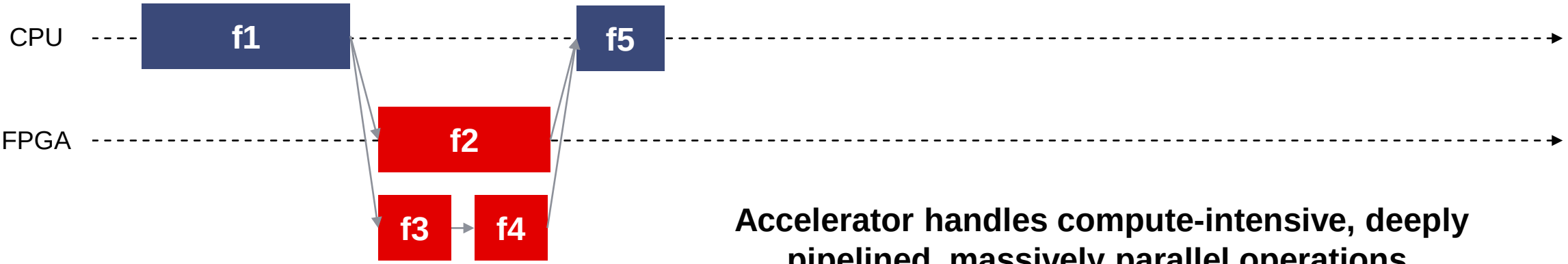
- Understand how to improve the performance of the system by using hardware accelerators
- How to identify functions for acceleration
- Be aware of overheads and limitations

Hardware Acceleration: Boosting Application Performance

f = function

Without acceleration – serial execution

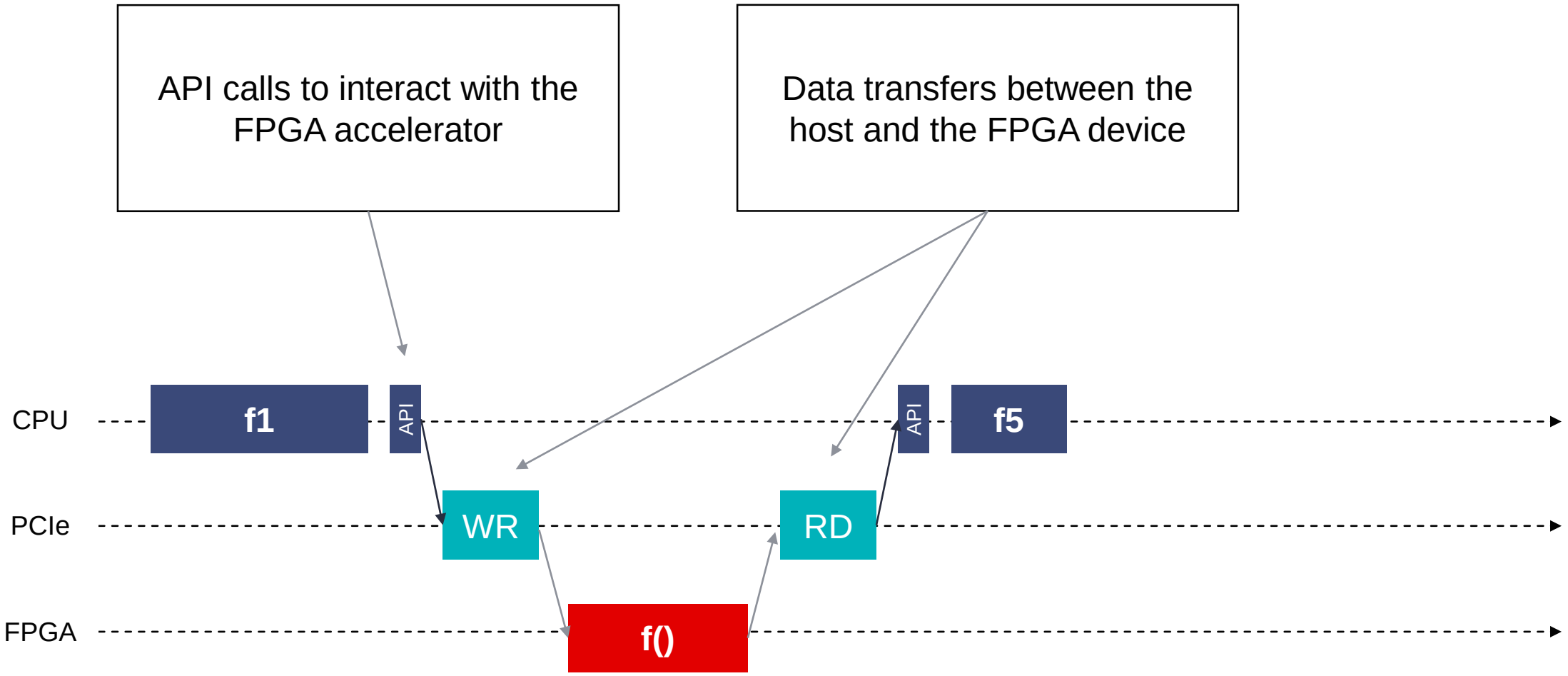
With hardware acceleration – parallel execution within and across functions



Accelerator handles compute-intensive, deeply pipelined, massively parallel operations.
CPU handles the rest

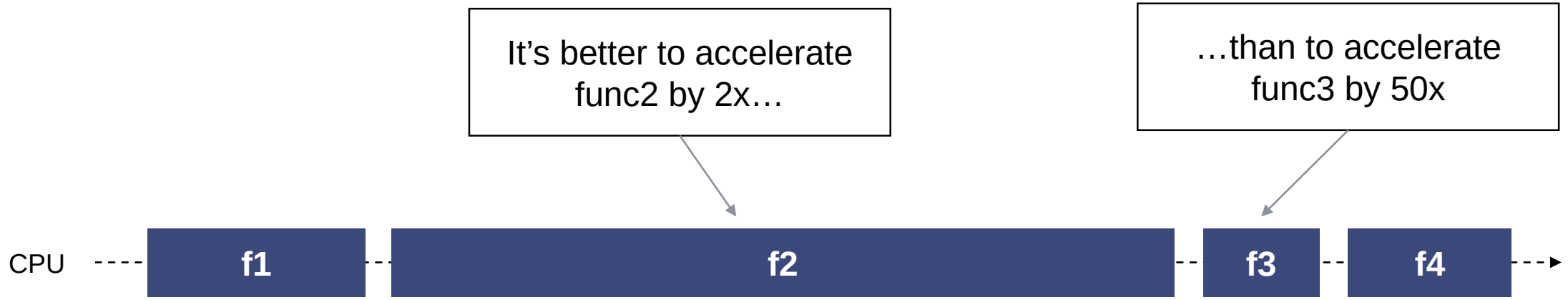
XILINX.

Hardware Acceleration: A More Accurate View



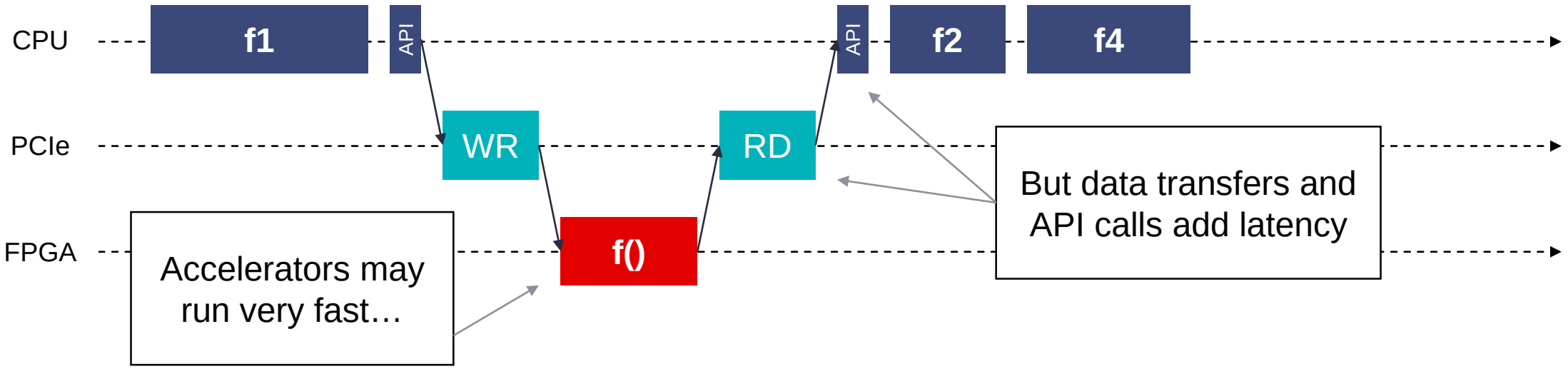
XILINX.

Boosting Application Performance



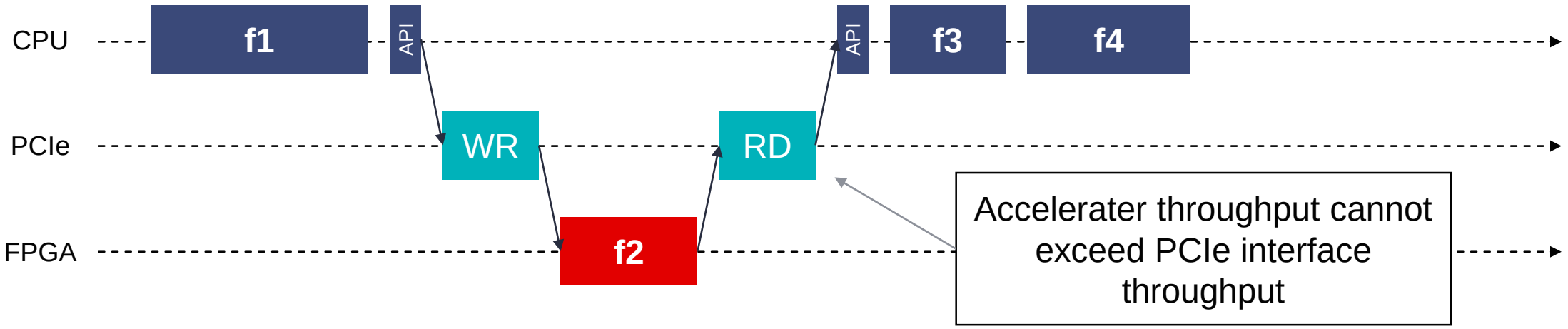
- Consider overall performance not just individual functions
- Target accelerators that will impact end-to-end performance of the application
- When working "top down", identify performance bottlenecks in the application
  - Use profiling tools, analyze the "roof line" of a flame graph

Identify Functions with Acceleration Potential



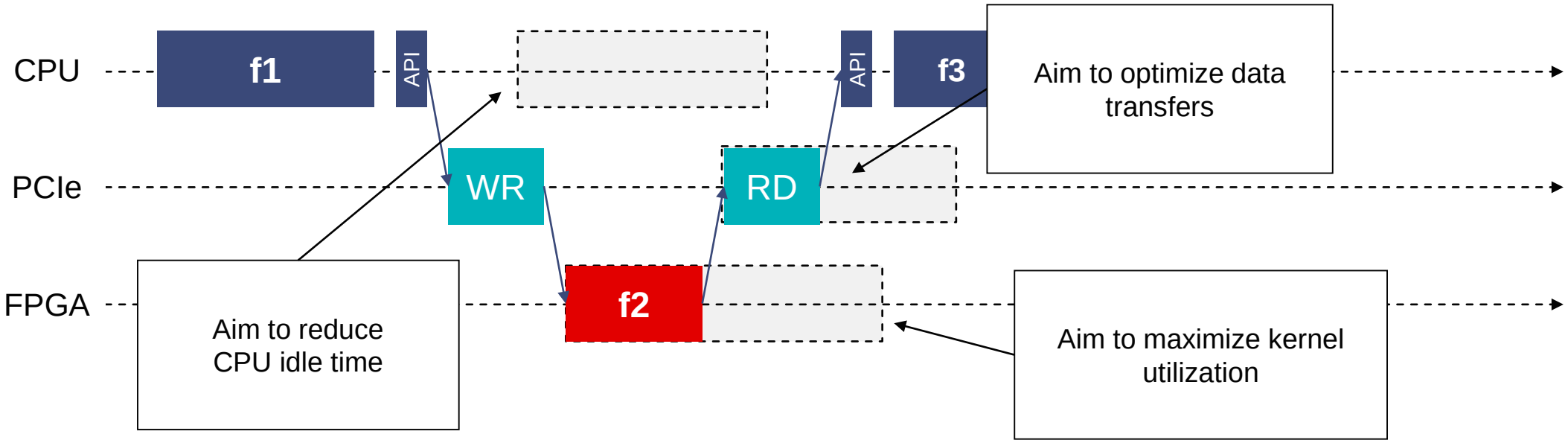
- Look for functions where {compute time} is much greater than {data transfer time}
  - Good: Monte Carlo – a few inputs, a lot of computations
  - Not so good: Vector addition – 2x more inputs than computations
- Functions that perform a lot of processing per invocation are preferable over small functions that are called many times
  - Minimizes API calls and event management overhead

Know the Constraints



- FPGA are highly configurable, but don't have infinite resources
- Start by understanding the characteristics of your target accelerator
- Know what will be the performance ceiling of your application

Architect the Software Application for Parallelism



- Minimize CPU idle time and do other tasks while the FPGA kernels are running
- Keep the kernels active, performing new computations as often as possible
- Optimize data transfers to and from the FPGA
- Threads and asynchronous programming are helpful to achieve this

Thank You