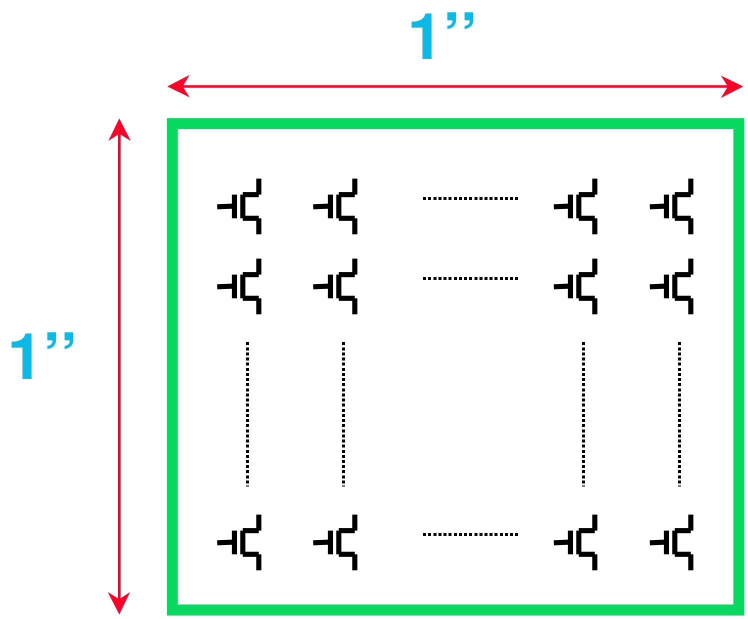
Why

CAD for VLSI → Software Compilation

The 100 Million Transistor Question



WHAT DO WE DO WITH THEM ?

HOW BEST CAN WE USE THEM TO SOLVE OUR COMPUTING PROBLEMS ??

Observation

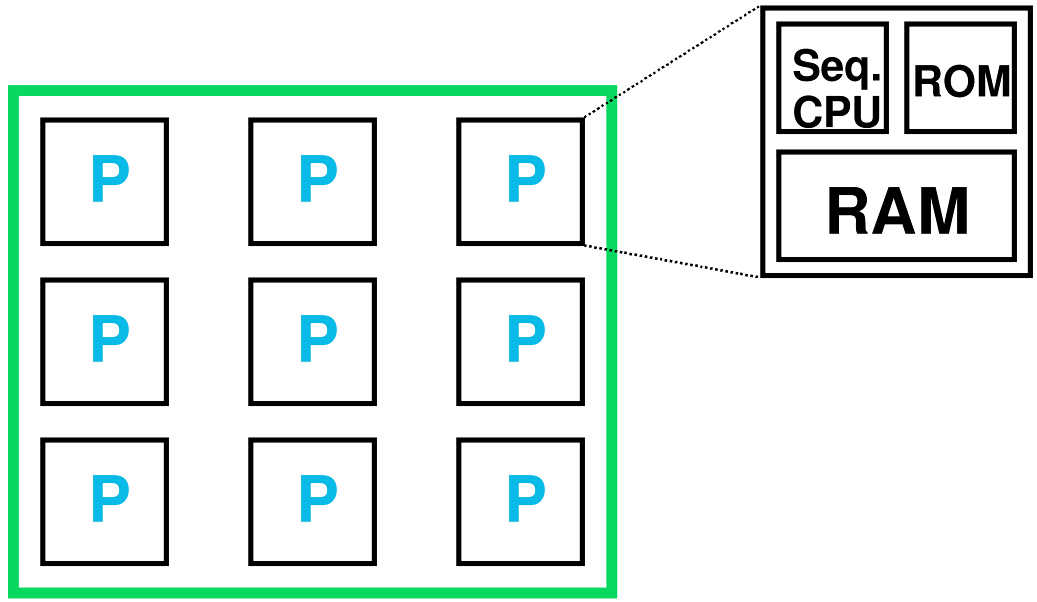
Whatever you put on the chip, you need a high-volume product to justify fabrication cost

Choices: programmability

configurability

~~Widely applicable "custom" chips~~

Answer I: Multiprocessor on a chip



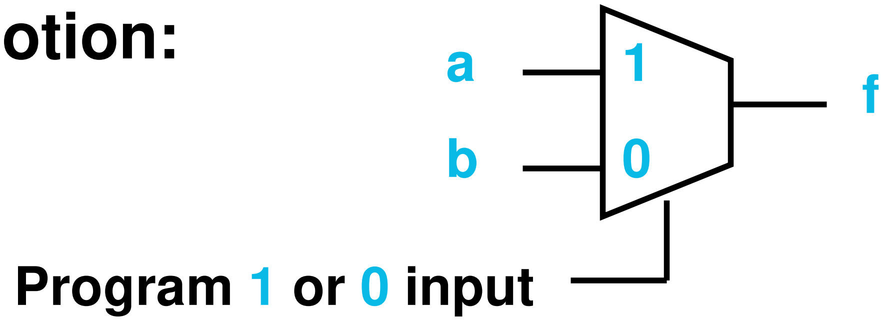
Requirement: Efficient, parallelizing compiler

Problems: "Enough" parallelism in programs?

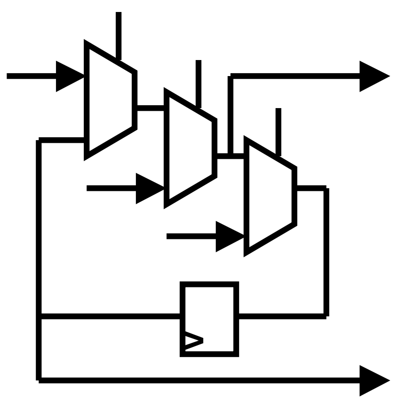
Doesn't go fast enough for video applications, for instance.

Configurable Logic

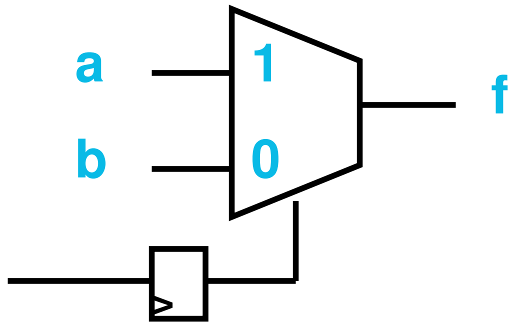
Basic notion:



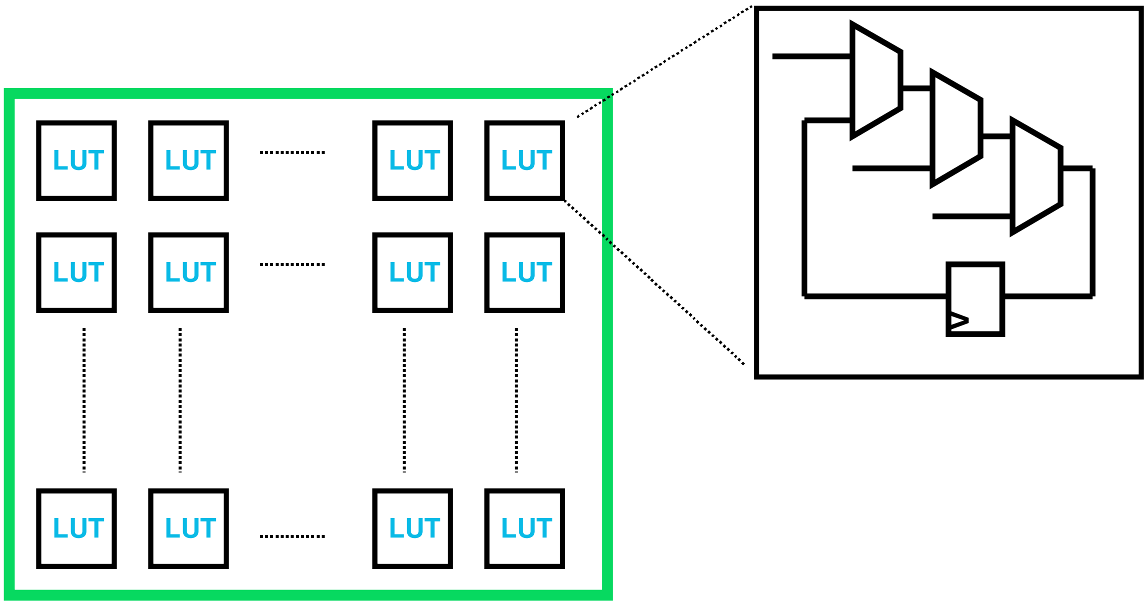
Compute element



Interconnect



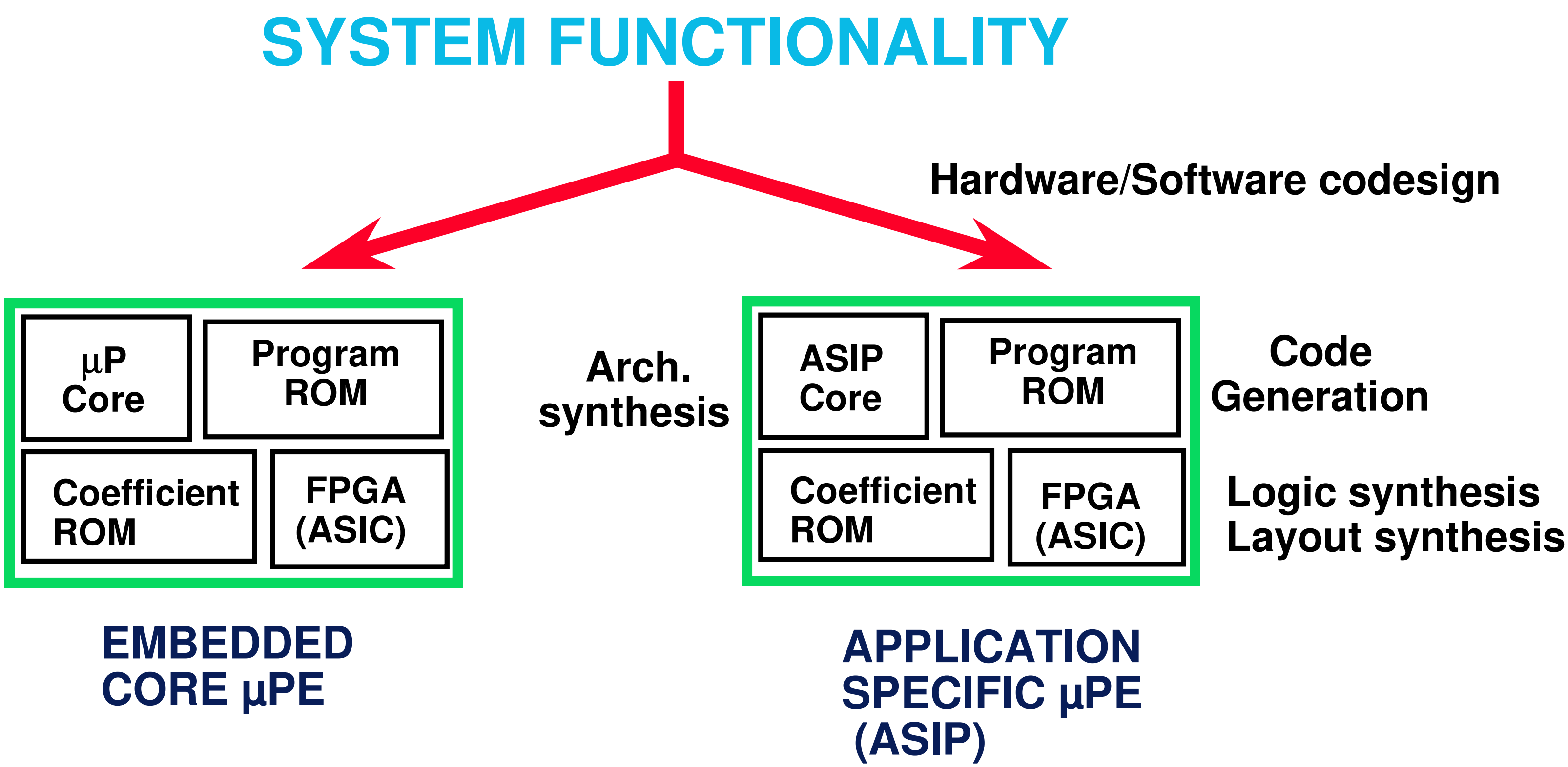
Answer II: Giant FPGA



Requirement: CAD system for FPGAs

Problems: May work well for bit-level video computations, but in general FPGAs are horribly inefficient

Everybody's Favorite Picture



Mixing Hardware and Software

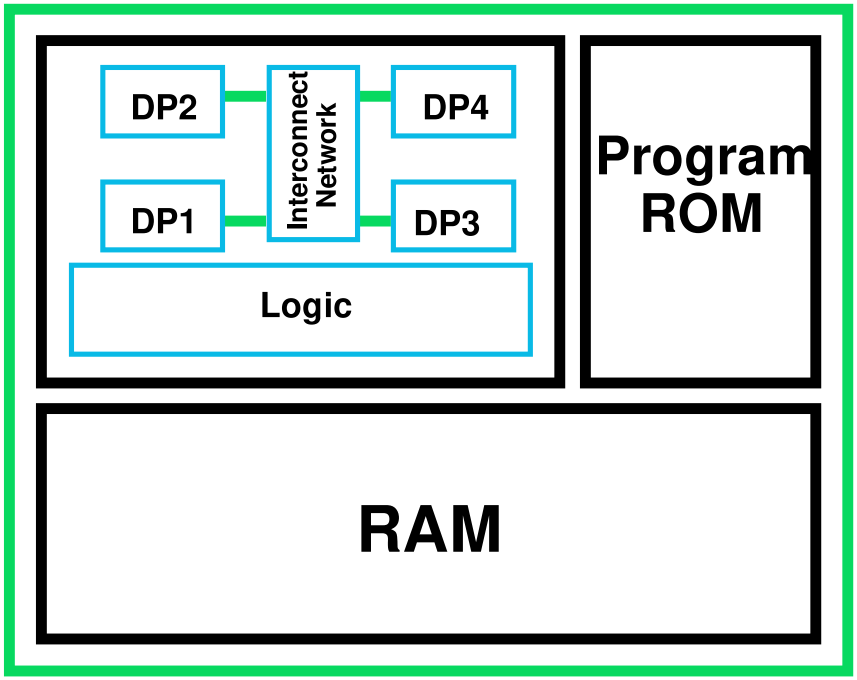
Argument: Mixed hardware/software systems represent the best of both worlds.

- High performance, flexibility, design reuse, etc.

Counterpoint: From a design standpoint, it is the <u>worst</u> of both worlds

- Problems of verification, and test become harder
- Too many tools, too many interactions, too much heterogeneity
- Hardware/software partitioning is “AI-complete” !

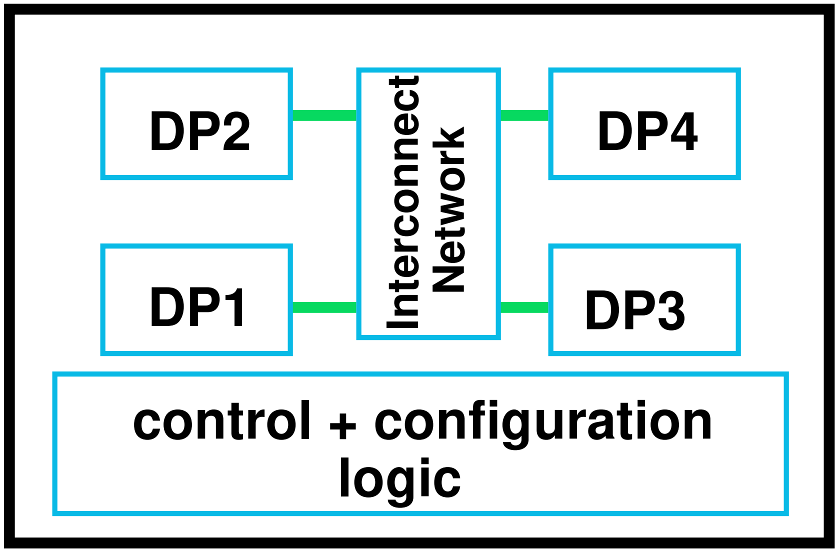
My new favorite ...



Throw out the ASIC/FPGA !

SIMD processor, many functional units, preferably VLIW, possibly superscalar

Reconfigurable Processor

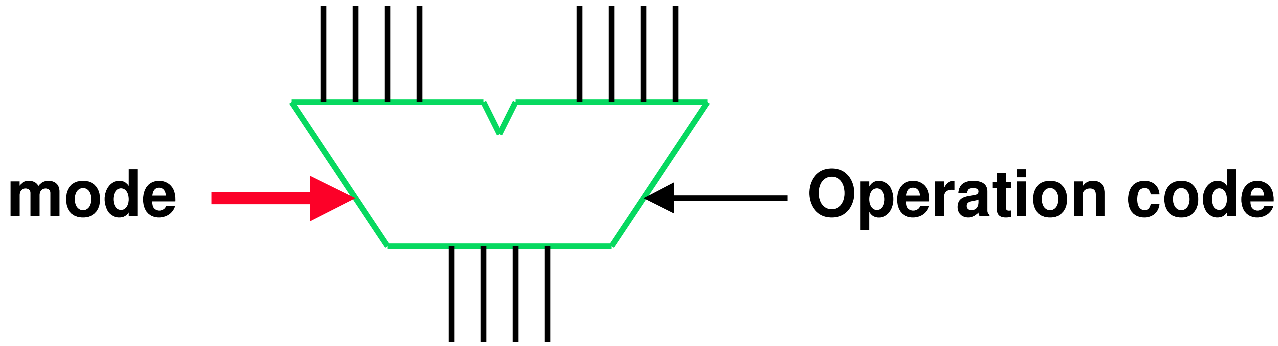


Configurable datapaths (e.g., splittable ALUs, complex operations)

Configurable interconnect (e.g., nearest neighbor, k buses)

Require <u>static</u> reconfigurability, dynamic optional

Configurable ALUs



Treat inputs as 32-bit operands, or two 16-bit operands, or four 8-bit operands

Configure to perform Boolean operations over more than two sets of inputs, e.g.,
F = A & B | C

Configured for different number representations

Processor Features

Processor can be high performance for a wide variety of applications because:

- \+ Tailor datapaths for particular application domains, or application, e.g., video compression
- \+ Functional units in datapaths can be heterogenous, performing differing sets of functions
- \+ VLIW will permit static scheduling, eliminating synchronization overheads

→ **ALL THE SMARTS IS IN THE COMPILER**

- – Synchronization of heterogenous operations in each instruction may impact performance adversely; may need superscalar architecture
- – May need dynamic reconfigurability

Code Generation and Optimization

Focus on automatic retargetability and parameterizable optimization methods

- Instruction selection for configurable functional units
- Scheduling for multiple functional units
- Register bank allocation to minimize data transfers
- Detailed register allocation with varying load/spill costs
- Optimization to exploit address generator features

and the list goes on …

Goal is to be able to generate high-quality code for any target architecture description

WHEN THAT HAPPENS, CAD IS GOING TO ESSENTIALLY BECOME SOFTWARE COMPILATION !!