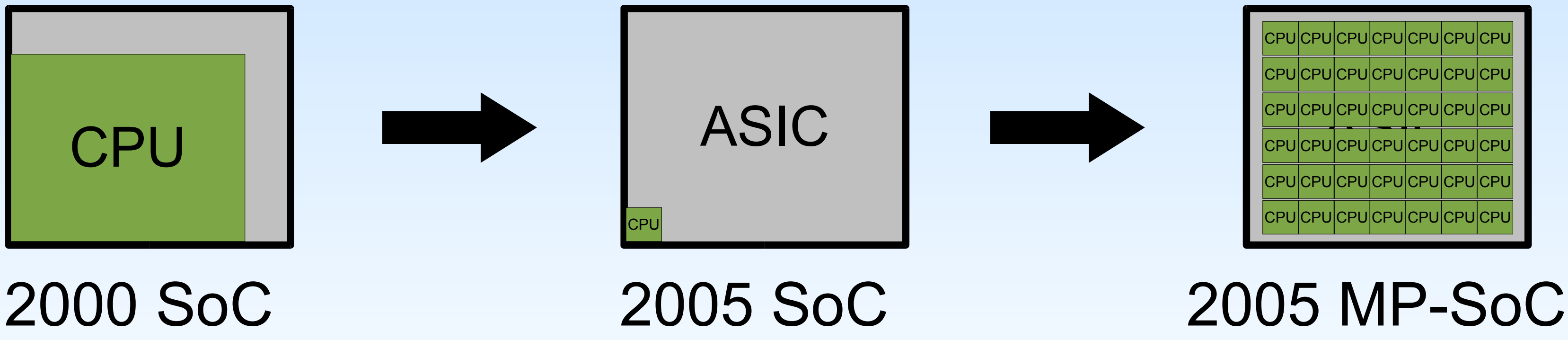
Programming Configurable Multiprocessors

Cmpware, Inc.

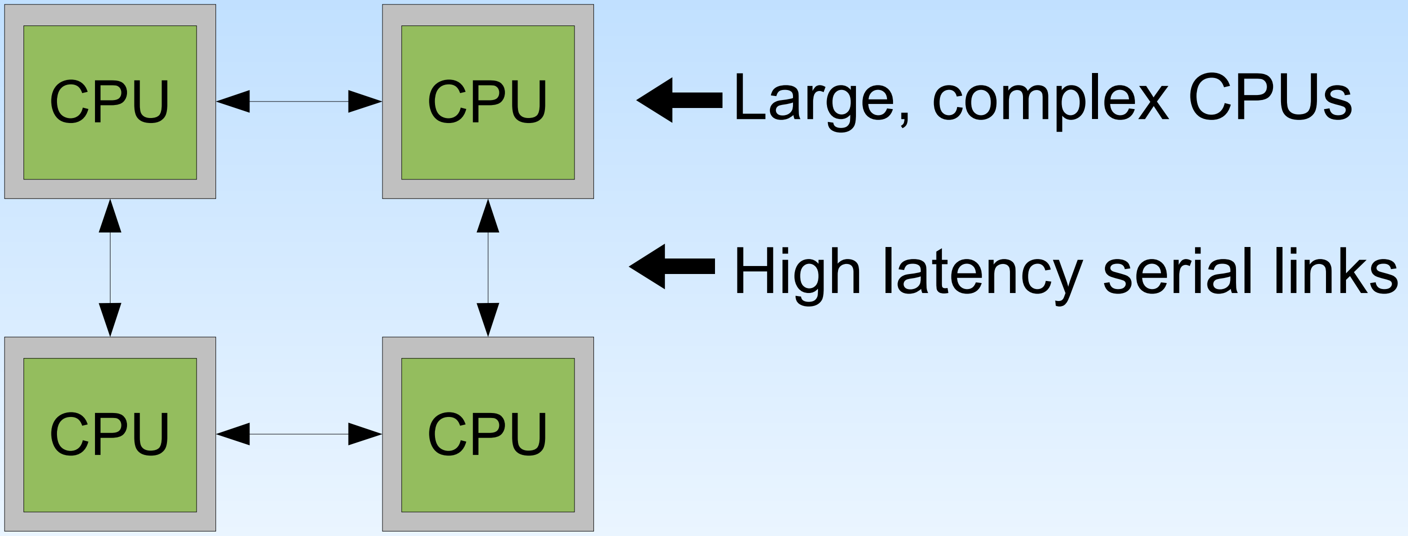
Introduction

- One billion transistors available (2005)
- 1B transistor custom designs very expensive
- Emerging trend: *multiprocessors*
  - Large, programmable IP blocks (CPUs)
  - Thousands of CPUs / millions of MIPS


CPU
2000 SoC
ASIC
CPU
2005 SoC
CPU CPU CPU CPU CPU CPU CPU
CPU CPU CPU CPU CPU CPU CPU
CPU CPU CPU CPU CPU CPU CPU
CPU CPU CPU CPU CPU CPU CPU
CPU CPU CPU CPU CPU CPU CPU
CPU CPU CPU CPU CPU CPU CPU
2005 MP-SoC

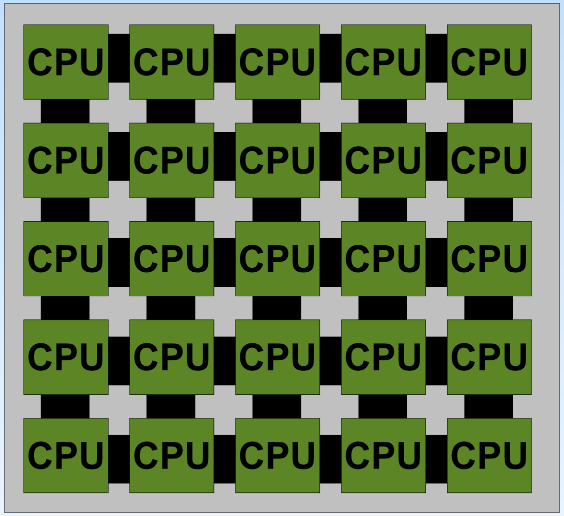
System Level Multiprocessors

- Dozens of complex processors
- Low communication / computation ratio
- Large grained parallelism (tasks)



Chip Level Multiprocessors

- Thousands of simple processors
- High communication / computation ratio
- Fine grained parallelism (sub-task)



← Small simple CPUs

← Low latency
parallel links

Hardware Design Programming Model

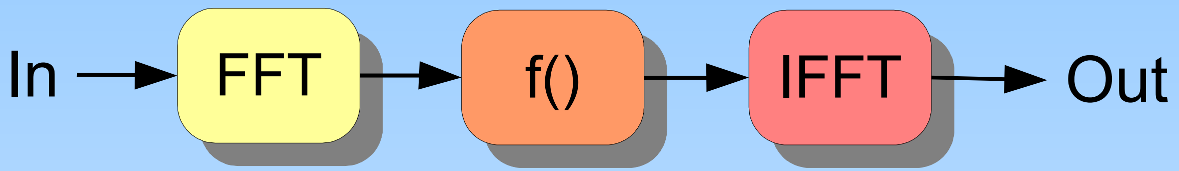
- A popular, well-understood model
- Increasingly similar to software (HDLs, ESL)
- Highly parallel / high performance
- Placement techniques ('floorplanning')
- *Restrictions*:
  - Array of CPUs, not 'random logic'
  - On-chip network, not 'random wires'
  - Resembles programmable logic devices

Programming Techniques

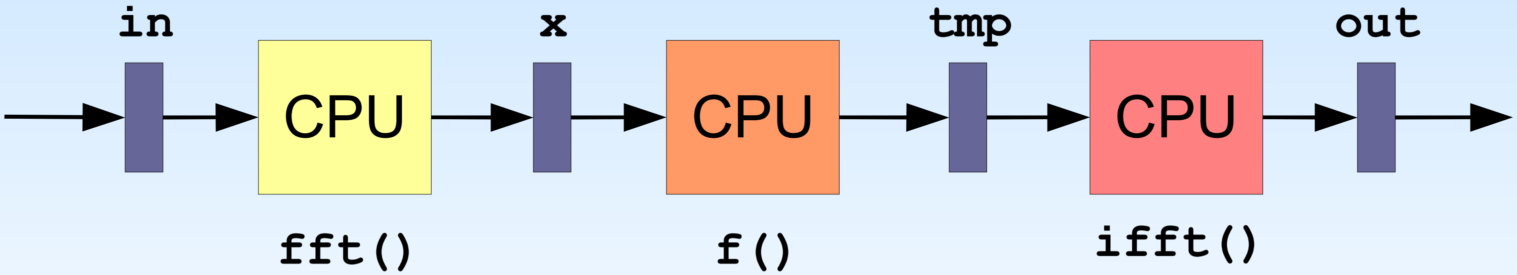
- Use traditional tools and languages ('C')
- Use **`volatile`** variables in shared memory
- Communicate via variable assignment ('=')
- Read shared variables *in* as parameters
- Write shared variables *out* as results

```
volatile int *in, *out;

*out = myCode(*in);
```

Filtering Example



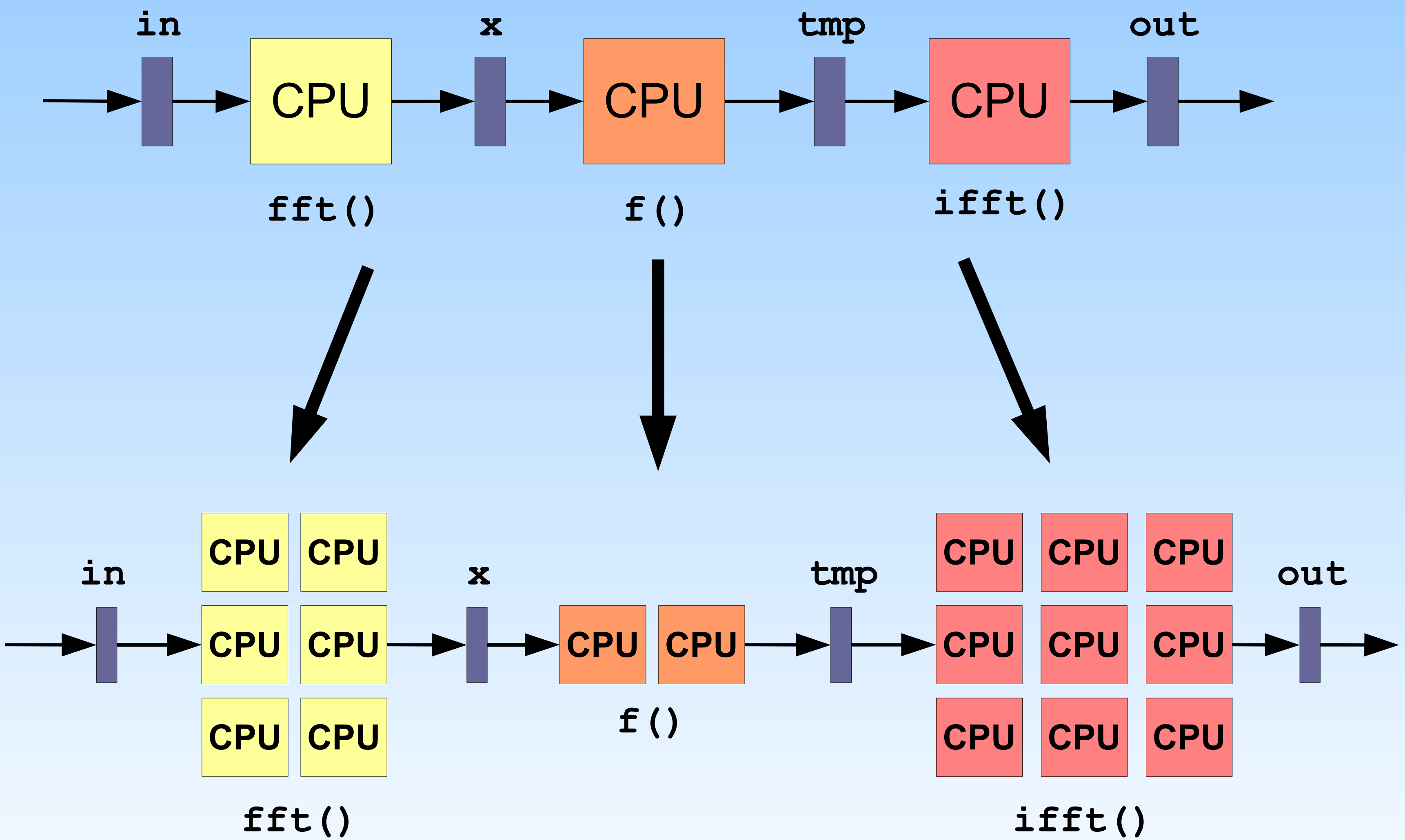
```
volatile int *x, *in, *tmp, *out;

*x = fft(*in);
*tmp = f(x);
*out = ifft(tmp);
```



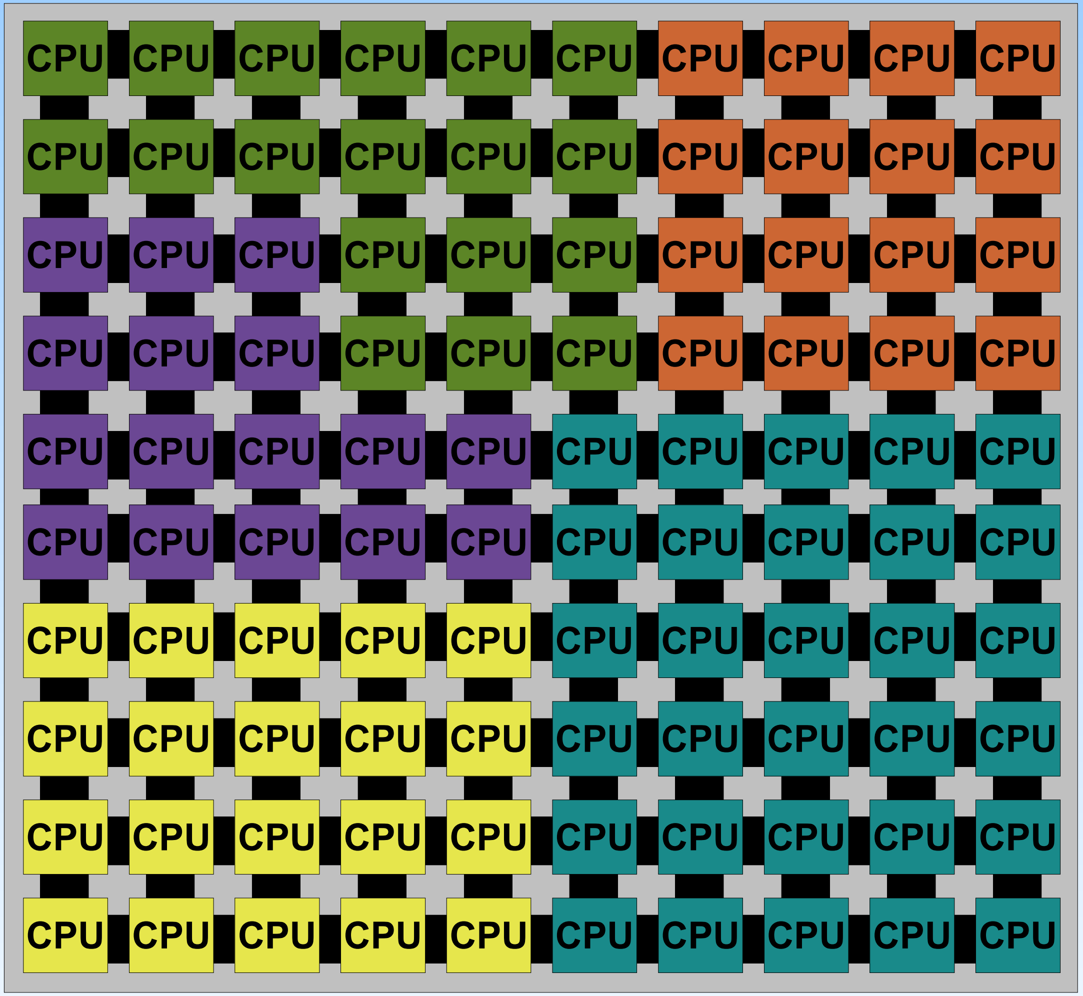
Exploiting Parallelism

- Sub-task / procedure level parallelism
- Break procedures down into smaller blocks
- Assign blocks to processors
- Replace local variables with shared volatiles
- Hardware technique: floorplanning
  - Group functions for efficiency
  - Group according to communication patterns
  - Groupings may be heirarchical

Sub-Task Parallelism



Software Floorplanning


CPU CPU CPU CPU CPU CPU CPU CPU CPU CPU
CPU CPU CPU CPU CPU CPU CPU CPU CPU CPU
CPU CPU CPU CPU CPU CPU CPU CPU CPU CPU
CPU CPU CPU CPU CPU CPU CPU CPU CPU CPU
CPU CPU CPU CPU CPU CPU CPU CPU CPU CPU
CPU CPU CPU CPU CPU CPU CPU CPU CPU CPU
CPU CPU CPU CPU CPU CPU CPU CPU CPU CPU
CPU CPU CPU CPU CPU CPU CPU CPU CPU CPU
CPU CPU CPU CPU CPU CPU CPU CPU CPU CPU
CPU CPU CPU CPU CPU CPU CPU CPU CPU CPU

Dynamic Behavior

- Flexiblity not possible with fixed hardware
- Add / change functionality
- A procedure call converts **`f(x)`** to **`z(y)`**
- Response to external inputs
  - Adjust functionality (*example*: MP3 ==> WMA)
  - Trade power consumption for performance
  - Reduce system size
- Highly reconfigurable

Conclusions

- Very high performance available through single chip multiprocessing
- A highly programmable solution
- Hardware-style programming techniques can:
  - Provide programmability
  - Leverage existing tools
  - Simplify porting of existing code
  - Give high levels of performance