CSSE4000/7002 Digital System Design II

Reconfigurable System on Chip

The story so far

- System on Chip
  - Modular (re)use of IP cores
    - CPUs, IO, memory interfaces, DSP
    - Buses and interconnect
  - Higher level abstractions
    - Design efficiency
    - Verification

The story so far

- Reconfigurable Computing
  - Avoid / reduce CPU inefficiencies
    - Computational density
    - Spatial vs temporal computing
  - FPGAs increasingly favourable implementation technology
    - Economics and performance
    - Flexibility and risk reduction

Reconfigurable System on Chip

- Implementation of SoC-like architectures on reconfigurable fabric
  - CPUs, memory, buses/interconnect
  - “Standard” IO and peripheral devices
  - Custom processing units or IO devices
  - Includes FPGAs with integrated hard CPU cores

Reconfigurable System on Chip

- Are these rSoC?
  - Commodity SoC with FPGA coprocessor
  - Workstation/PC with PCI/X FPGA processing card
  - Custom DSP processing engine implemented in FPGA

Best of both worlds?

- Temporal computing when you can get away with it
  - TCP/IP stack and webserver in VHDL?
- Spatial computing when you need it
  - 2048-tap FIR filter @ 1M sample/sec in SW?
- Time to market / time in market

Worst of both worlds?

- Power consumption
  - FPGAs power hungry, but getting better
  - Still no match for low-power SoC devices
- Design tools
  - SoC platform tools improving
  - RC tools still fairly primitive
- Unit costs
  - There is still a crossover point

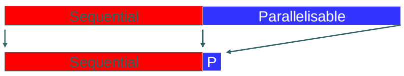
Use models for RSoC

- Application acceleration
  - Custom data processing
  - IO preprocessing
- Operating system support
  - HW scheduler
  - HW context switch
  - Semaphores/mutual exclusion

Amdahl's Law

- Speedups come from exploiting parallelism
  - Algorithms usually have serial and parallel phases
  - Maximum possible speedup determined by the parallelisable fraction *P* (or serial fraction *S: S+P=1)*
  - .

$$Speedup_{max} = \frac{1}{S - \frac{(1-S)}{N}}$$

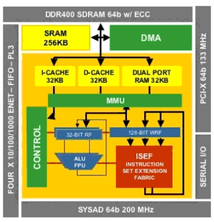


If *P* is 50%, maximum possible speedup is 2 (twice as fast)

Coupling

- Tight coupling
  - Close interaction between control flow and data flow
  - Units operate in lock-step (closely synchronised)
  - Typically fine-grained operations
- Loose coupling
  - Weak interaction between control and data flow
  - Units operate more independently
  - Typically coarser-grained

Architectures – custom instructions / register mapped

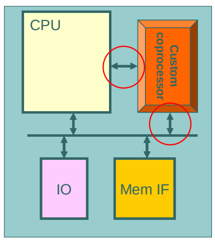


[Stretch, Inc.]

- Tight coupling between SW and HW
- Must match CPU dataflow model
  - Register widths, pipeline stages
  - Simplifies interfacing
- Fine-grained parallelism
- Algorithm control flow remains in SW
  - Late binding
- Amdahl's Law limits practical speedups
  - How many custom instructions?
- Compiler defines HW/SW interface

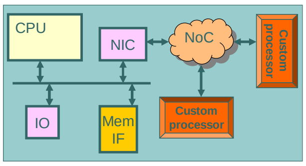
Architectures – coprocessors / bus mapped

Bus or direct connect



- Looser coupling between SW and HW
- HW less dependent on CPU architecture
- Control flow independent of CPU
  - Coarser grained
  - Early binding
- All actions still require CPU handling
- Potential for shared bus contention
- CPU must still handle all data
  - DMA can help
- SW libraries and bus protocol define HW/SW interface

Architectures – network mapped



- Very loose coupling
- Control flow independent of CPU
- Higher latency
  - Requires coarse granularity for efficiency
- More difficult to develop and program
- OS and network protocol define HW/SW interface

Architectures - comparison

Mapping	Strategy	Configuration mechanism	Coupling	Integration technology
Register mapped	Custom data path	Custom instruction	Tightly synchronised	Custom compiler
Memory / bus mapped	Coprocessor	Memory mapped instruction	Loosely synchronised	Software libraries
Network mapped	Peer processor	Configuration packets	Uncoupled	Network protocol

Some design cases to consider

- What architectures might be suitable for the following, and why?
- TCP/IP checksum offload
  - Incoming packets (up to 1500 bytes) must have the checksum tested to detect corruption during transmission
  - Outgoing packets have checksum computed and inserted in header
  - Computation is a simple logical operation, iterated over the entire packet
- AES encryption
  - Data is encrypted in 128-bit blocks, using 128-bit keys, producing 128 bits of encrypted cyphertext
  - Algorithm is multiple (e.g. 10) rounds of 4 steps
    - Logical XORing of two 16-byte tables (key and data)
    - Lookup-based substitution of resulting 16 bytes
    - Shifting of values within table
    - Multiple XORing of table entries against fixed values

Looking back

- Remember this?
  - Demonstrate understanding of
    - concepts of system on chip, reconfigurable computing and reconfigurable system on chip
    - Economic factors in rSoC design
    - phases of the design process
    - tradeoffs vs ASIC / standard cell / full custom
    - building blocks of rSoC architecture
    - architectural issues in rSoC architecture

A little about PWM

A little about PWM