MANCHESTER
1824

The University
of Manchester

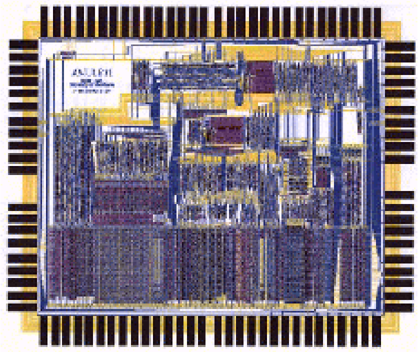
The Amulet Chips:
Architectural Development for Asynchronous Microprocessors

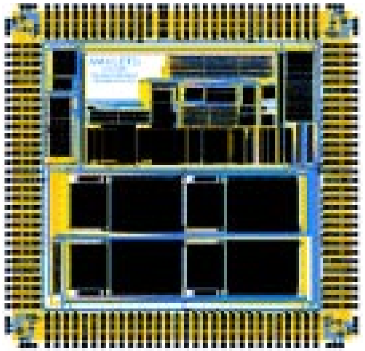
or

Some experiences of building large asynchronous systems

Amulet timeline

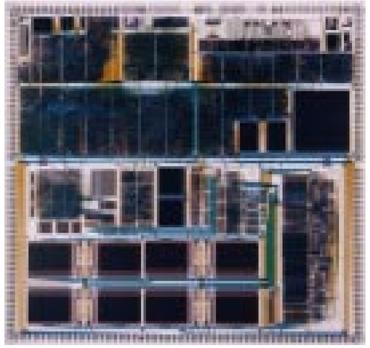
- Amulet1 1993 1.0 μm
- ‘ARM6’ microprocessor





- Amulet2 1996 0.5 μm
- ‘ARM7’ processor + cache + interfaces

- Amulet3 2000 0.35 μm
- ‘ARM9’ Processor SoC
- async. bus, RAM, ROM, DMA, DRAM interface
- … + synchronous peripherals



This talk …

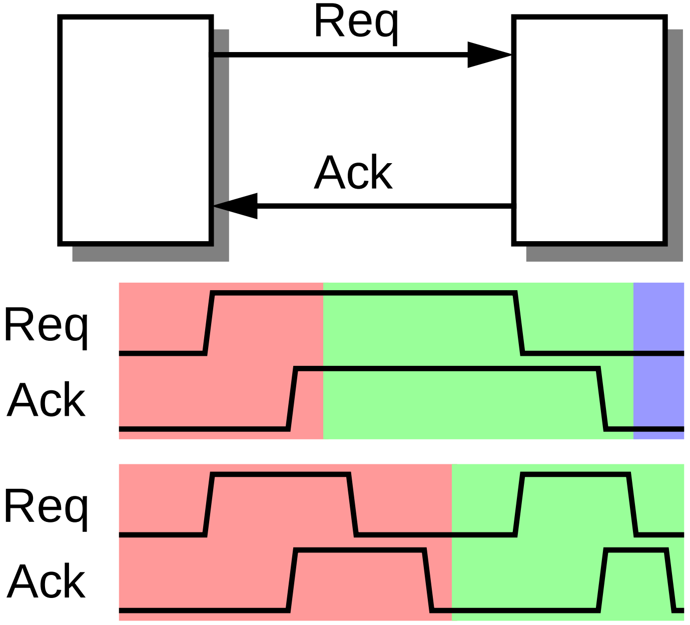
- Amulet processors are history:
    - ‘hard’ processor cores
    - local timing assumptions in implementation
    - engineering detail no longer relevant
- Attempt to highlight still-useful aspects
    - microarchitectural features
    - implementation choices
- How it has influenced our subsequent work

Some ‘differences’ with asynchronous design

- Pipelining is easy – perhaps too easy?
- Long-range interaction is difficult
- Stage timing can be data-dependent
- System may behave non-deterministically

Two-phase or four-phase?

Without a clock, systems communicate via handshakes:



Handshakes can be edges (two-phase) …

- faster, lower power

… or levels (four-phase)

- logic much simpler

- Amulet1 used two-phase signalling on and off chip
- Subsequent devices used four-phase signalling

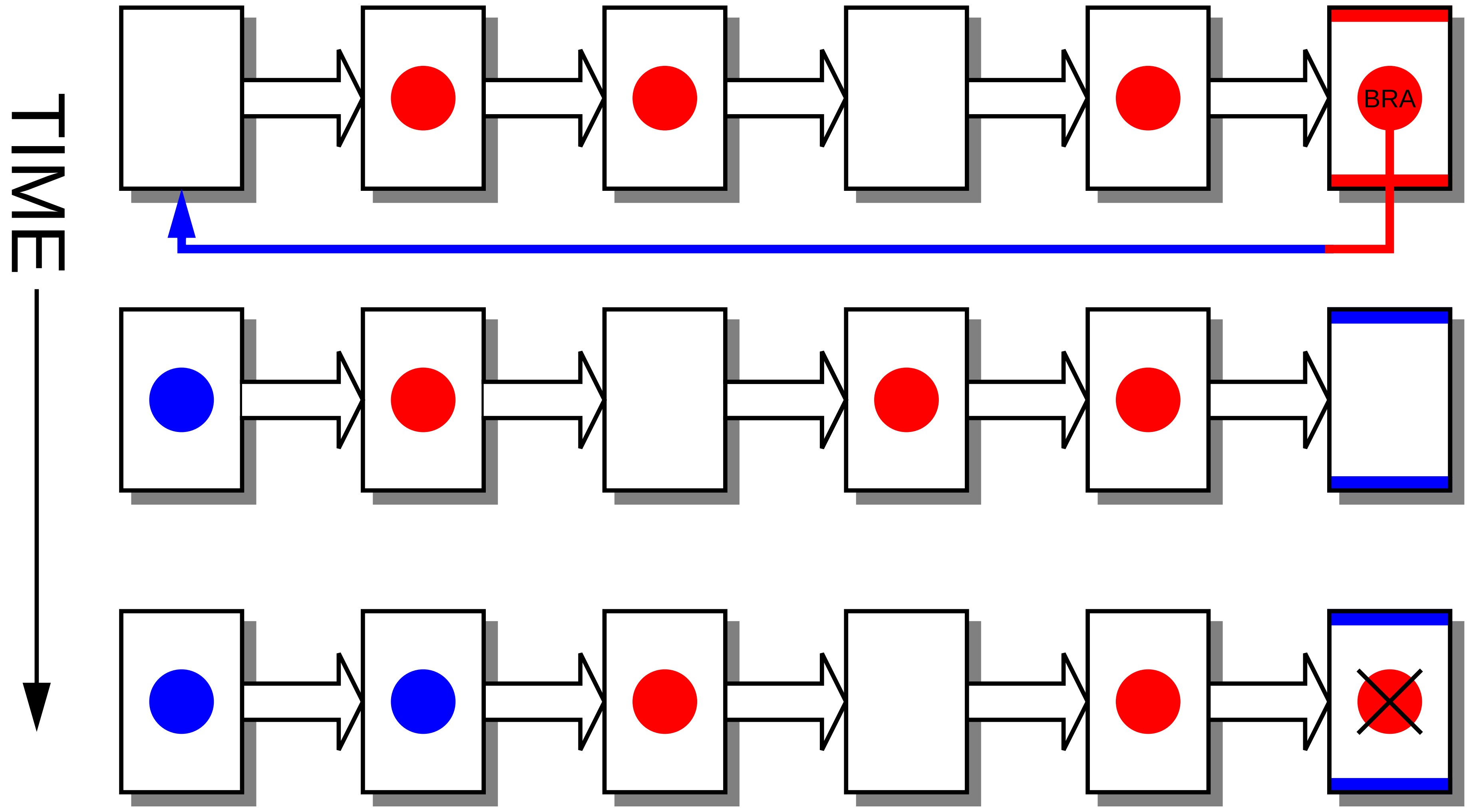
But for a recent, GALS device we have used two-phase for inter-chip communication

Flushing a pipeline

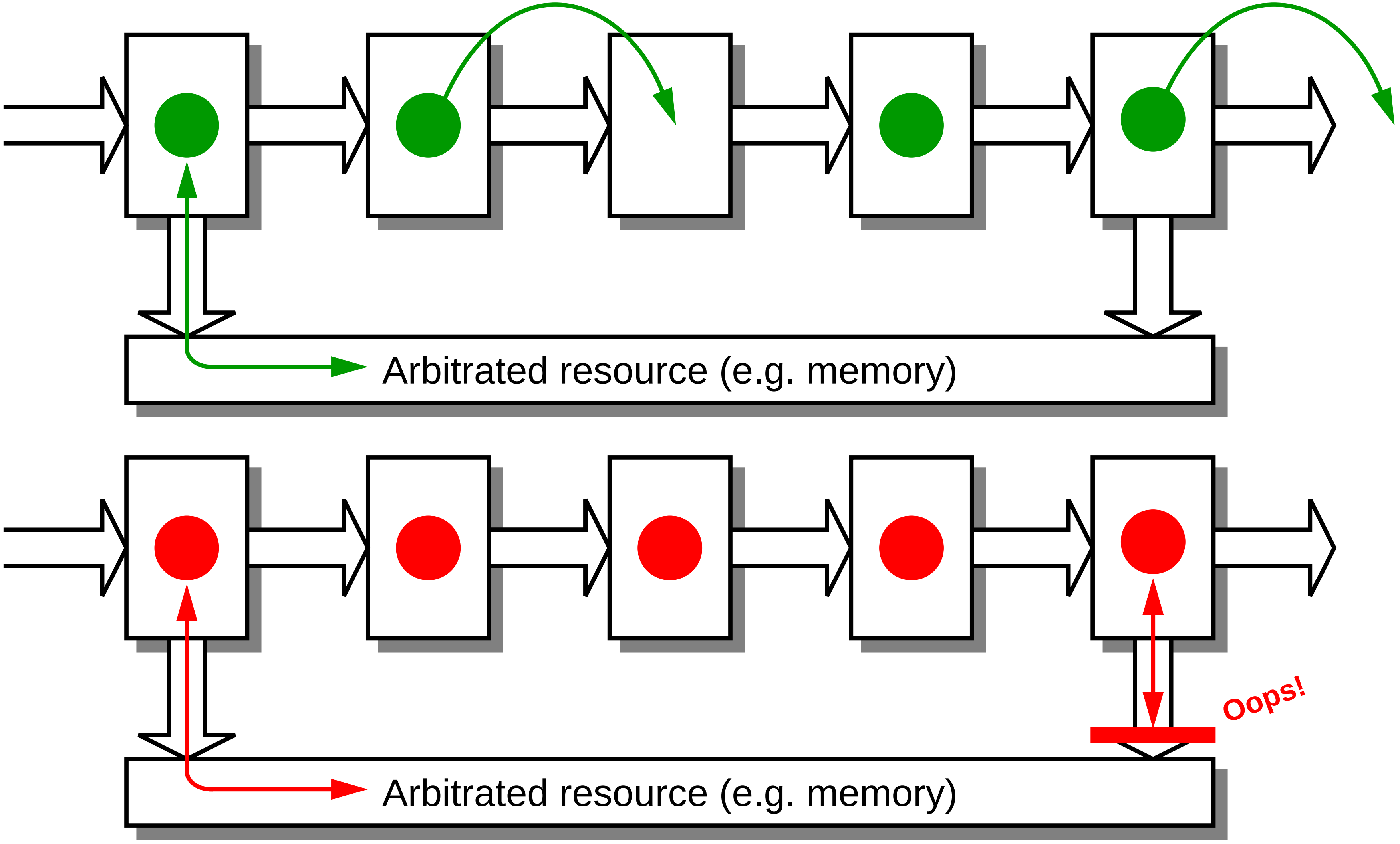
- Problem: prefetch has fetched unwanted instructions
    - possibly an unknown number
- Cannot discard for a known number of cycles

- Solution: ‘colour’ the instruction flows
    - committing to a branch means the current colour is no longer accepted
    - stream from branch target fetched in a different (acceptable) colour
    - two colours adequate in simple system

- Extensible
    - Amulet3 uses another colour to allow late data aborts


TIME
BRA

Deadlocks


Arbitrated resource (e.g. memory)
Arbitrated resource (e.g. memory)
Oops!

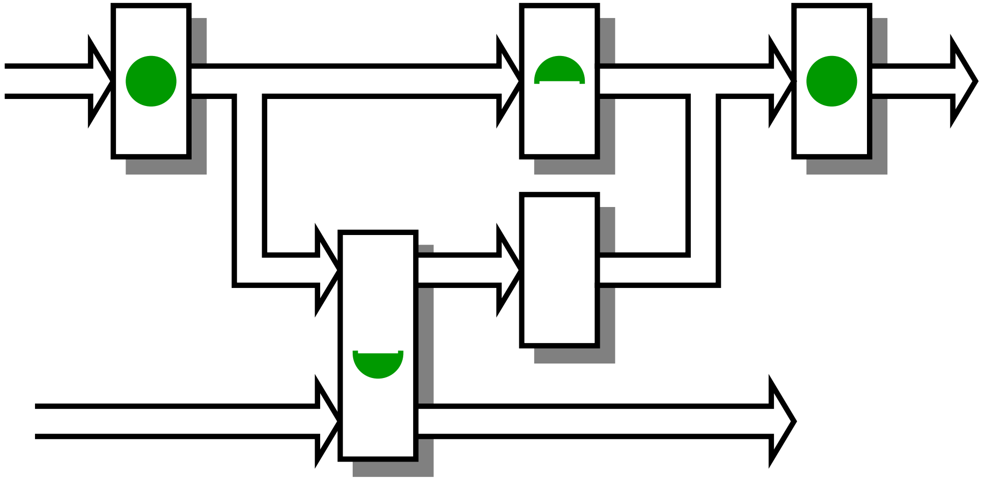
Deadlocks

In the preceding example:

- A deadlock can occur … but only in certain circumstances
  - hard to reproduce in simulation
- The reachable state space can be very large
  - much larger than in a synchronous machine
  - too large for formal tools?

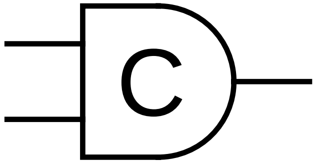
Rule: don’t start a ‘process’ (‘lock’ a unit) until it’s guaranteed that it will complete

Mechanism parallel token pipes act as ‘throttle’



Synchronisation

- A clocked system relies on the clock to synchronise data transfers
- An asynchronous system synchronises locally
  - transmitter is ready to transmit
  - receiver is ready to receive
  - one-to-one correspondence is maintained



Muller C-element

- Sometimes it is not predictable whether a communication will be required

Example:

```
add    r2, r1, r0
sub    r4, r3, r2    ; dependency on preceding instruction
```

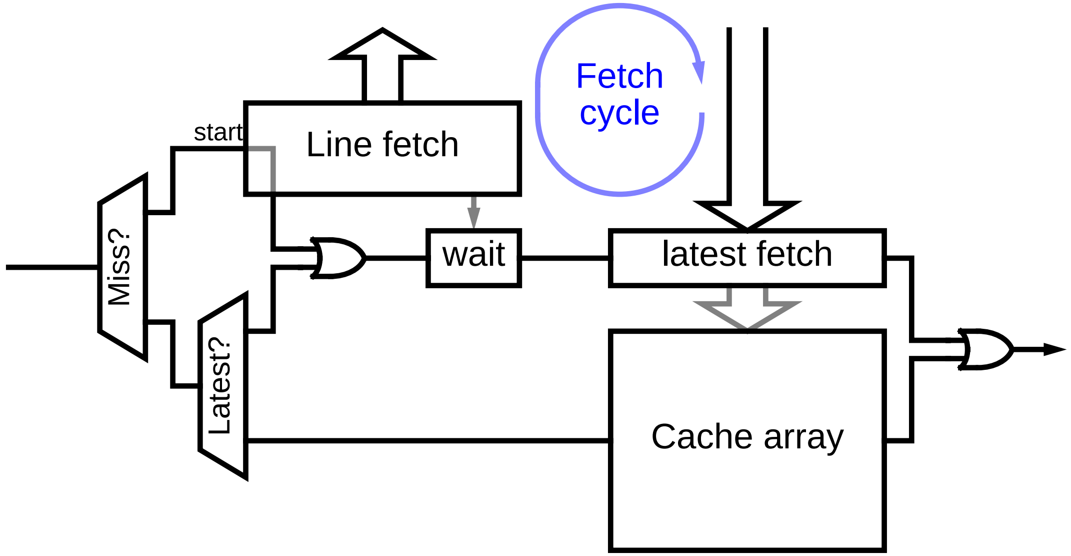
  - The `add` is not aware that it should forward its result when it is issued
  - Forwarding occurs to ‘zero or more’ following instructions

Synchronisation

Another form of synchronisation can be provided with a simple AND gate

In a defined interval:

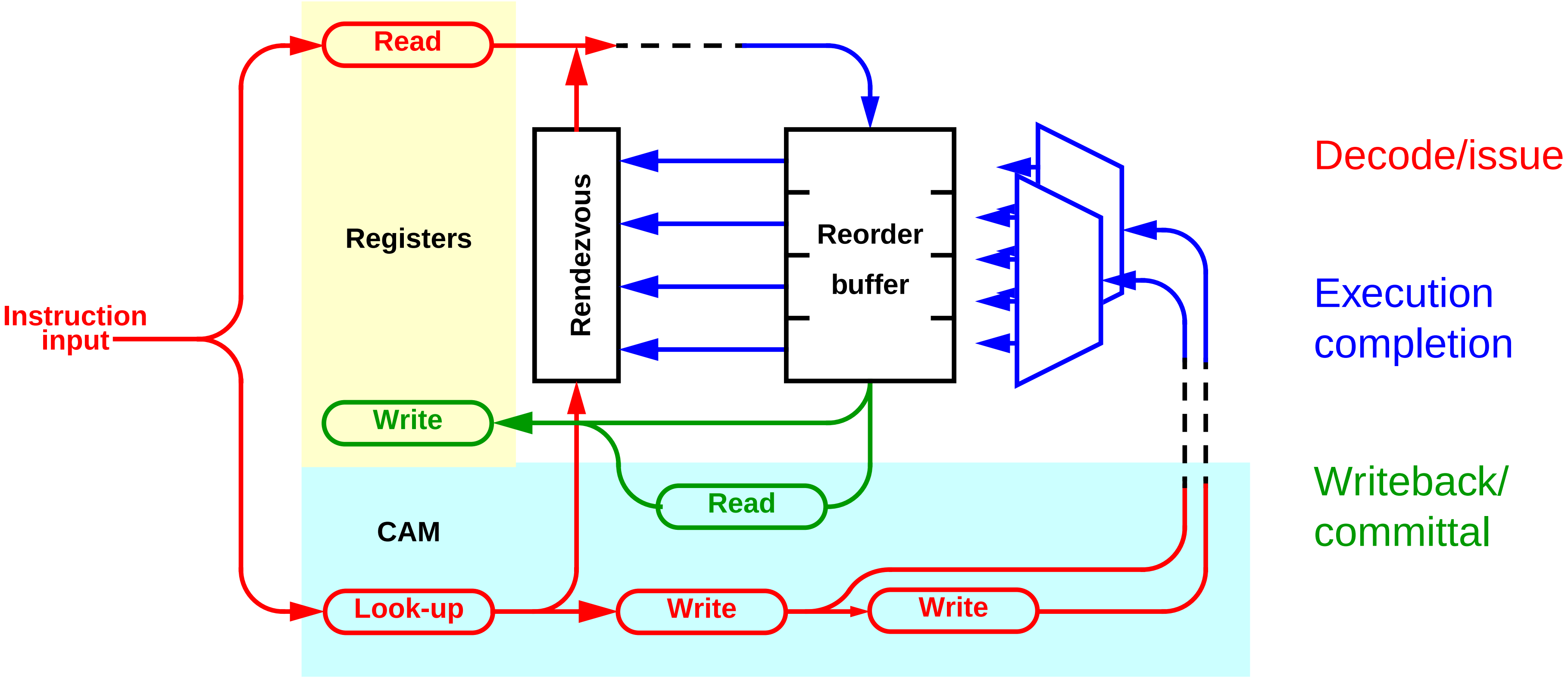
- IF one input can be guaranteed to change monotonically 0⇒1 …
- … any number of handshakes on the other input may be attempted
  - the first *may be* delayed



Example: reading words from a cache line concurrently being fetched (Amulet2)

Reorder buffer

Example, using many of the preceding techniques



- The different stages are only synchronised when necessary
  - e.g. forwarding and writeback are independent

More recent work …

- Asynchronous language development continues
    - ‘Teak’ offers better (“more conventional”) asynchronous circuit synthesis

- Processor development using Balsa synthesiser
    - “nanoSPA” achieves ~50% ARM performance

- GALS networking on- and off-chip
    - Silistix networking used in new SpiNNaker chip
    - Low-power, noise tolerant, two-phase off-chip links

Conclusions

- Can duplicate sophisticated architectural functions without a clock
  - similar architectures give similar performance to synchronous chips
- Deadlocks are a threat
  - requires careful state-space analysis
- Design process has similarities with asynchronous software
  - such as operating system development
- Four-phase good, two-phase bad
  - (mostly)

- There are real opportunities, especially for EMC
- For fine-grain, super-fast pipelines in processes with high manufacturing variability there may be no alternative!

Acknowledgements

- The whole Amulet team throughout the decade of development
  - too numerous to list here

- The European Commission
  - much of this work was funded by various projects from the Open Microprocessor Systems Initiative

Questions?