AMULET3i
– an Asynchronous System-on-Chip

or

Adventures in self-timed microprocessors

"The time is out of joint; O cursed spite"

William Shakespeare

"For the times they are a'changin'"

Bob Dylan

www.cs.man.ac.uk/amulet/projects/AMULET3i.html

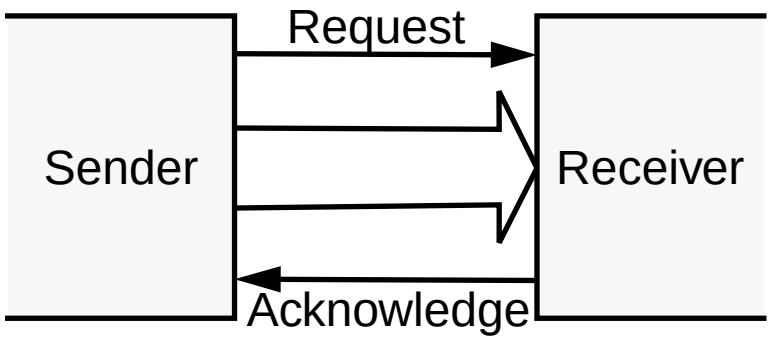
AMULET group

Why Asynchronous Logic?

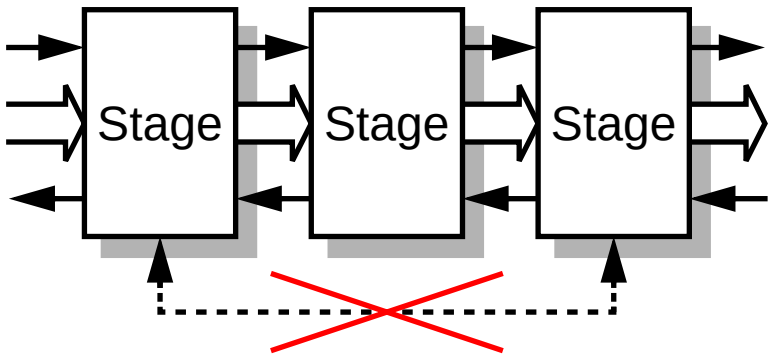
- Low power
  - Do nothing when there is nothing to be done
- Modularity
  - Added design freedom and component reusability
- Electromagnetic Interference (EMI)
  - Clocks concentrate noise energy at particular frequencies
- Security?
  - Surprise the hackers
- Crazy idea
  - Very few other people ~~are~~ *were* doing it

How is it done?

By handshaking …



- Message passing
- Rendezvous only when necessary



- Form into pipelines
- Non-local action disallowed (mostly)

What's hard about it?

Synchronous design allows:

- Static design of logic
- Non-local interactions
- Big choice of design tools
- Slowing down the clock if timing errors made

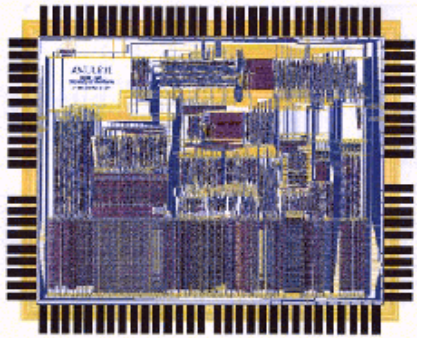
Asynchronous logic works well only for local interactions

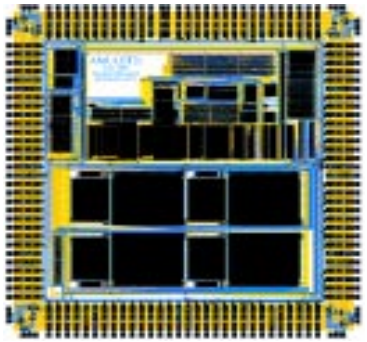
- Individual blocks of logic are straightforward
  - Timing must be carefully accounted for (self-timing)
- Simple pipelines are easy
- Interactions between 'distant' blocks hard (e.g. result forwarding)
  - Many synchronous structures "impossible" :-)
- Lack of suitable design/verification tools

What have we done?

AMULET1 (1994)

- ARM6 compatible processor (almost)
- Feasibility study
- 1.0 µm 60 000 transistors



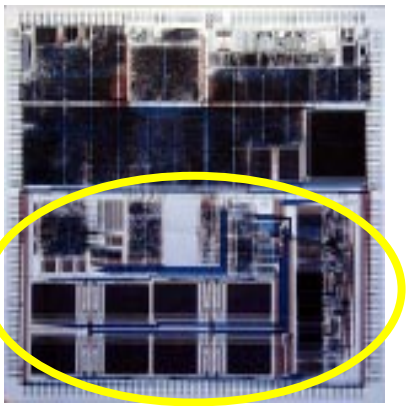


AMULET2e (1996)

- ARM7 compatible processor
- Asynchronous cache
- 0.5 µm 450 000 transistors

AMULET3i (2000)

- ARM9 compatible processor
- Memory, DMA controller, bus, …
- 0.35 µm 800 000 transistors
- Commercial application

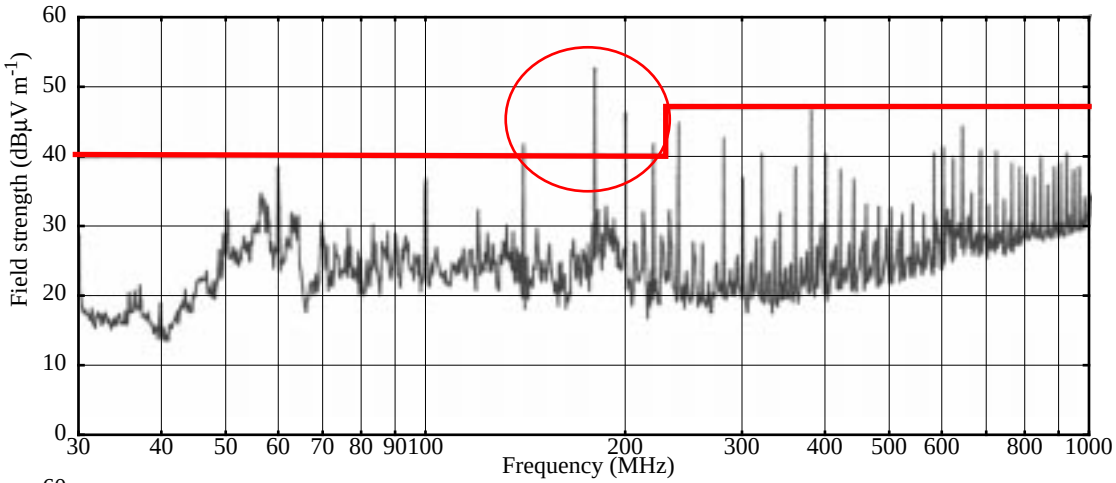


THE UNIVERSITY of MANCHESTER

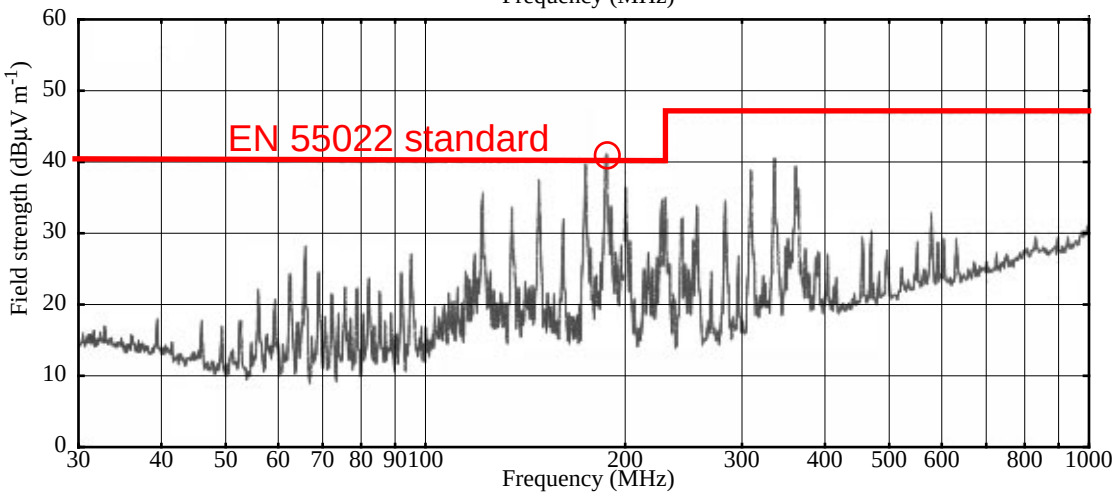
AMULET group

AMULET2e EMC

ARM6

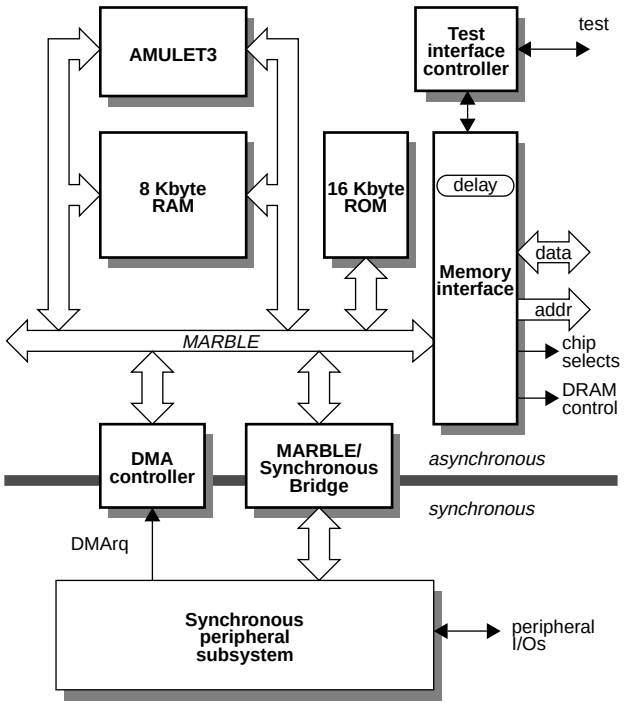


AMULET2e



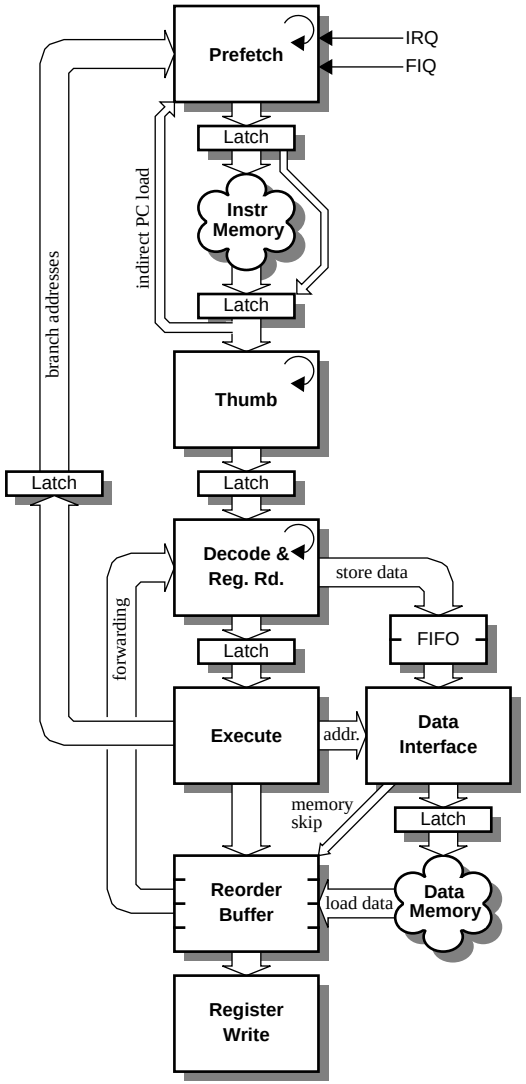
Enough justification for an AMULET3 product.

AMULET3i
- an Asynchronous System-on-Chip



- AMULET3 microprocessor (ARMv4T)
- 8 Kbytes RAM
- 16 Kbytes ROM
- Flexible multi-channel DMA controller
- Programmable external memory interface
- MARBLE, a fully asynchronous on-chip bus
- Bridge to on-chip synchronous bus
- Configuration registers
- Software debug support
- Test interface

AMULET3 Processor

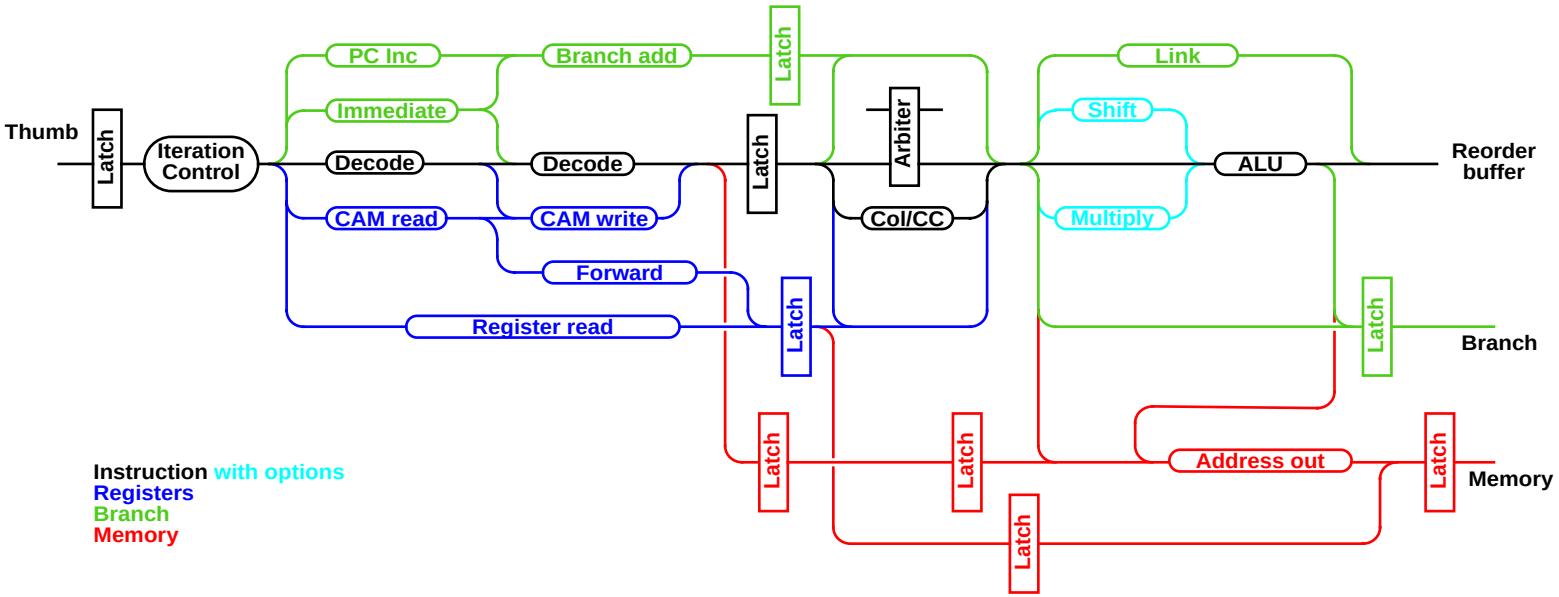


- Branch prediction
- Unwanted cycle suppression
- Automatic halt mode
- Thumb decoder
- Unrestricted register forwarding
- Load/store with out-of-order completion
- Dual (“Harvard”) bus interface
- Support for precise exceptions

THE UNIVERSITY of MANCHESTER

AMULET group

Process-level Parallelism

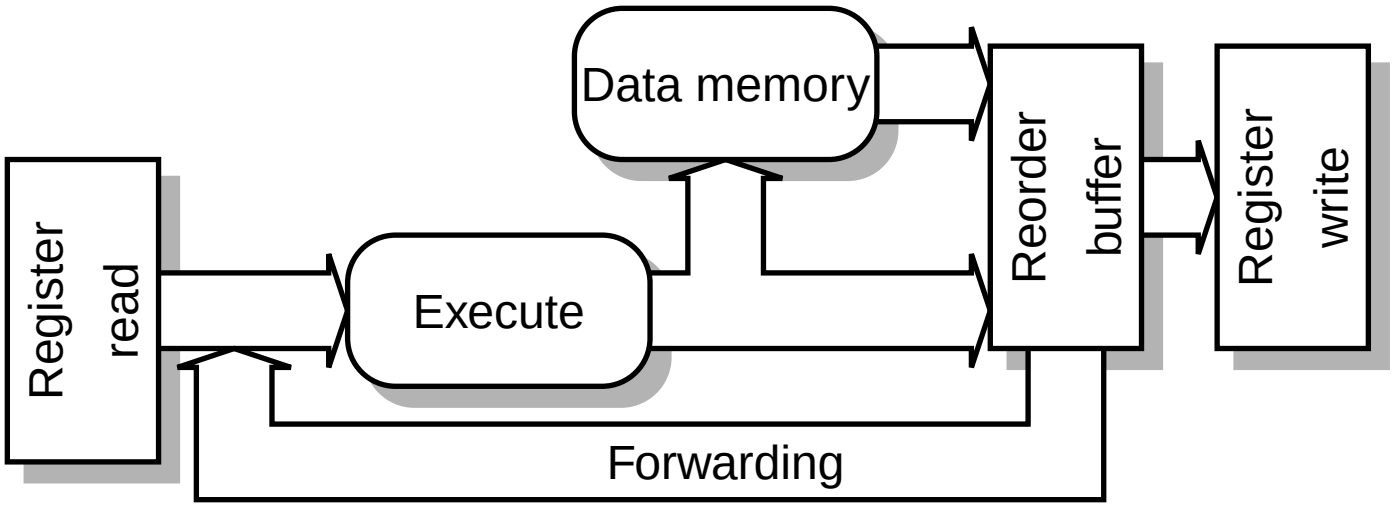


Example: decode & execute stages

- Various threads – many invoked conditionally
- Skewed pipeline latches (to lower power & EMI)
- Variable stage delay (e.g. ‘stretching’ cycle for series shift)
- Differing pipeline depths (extra buffer for LDM/STM)

Reorder Buffer

The reorder buffer is a key feature of AMULET3.

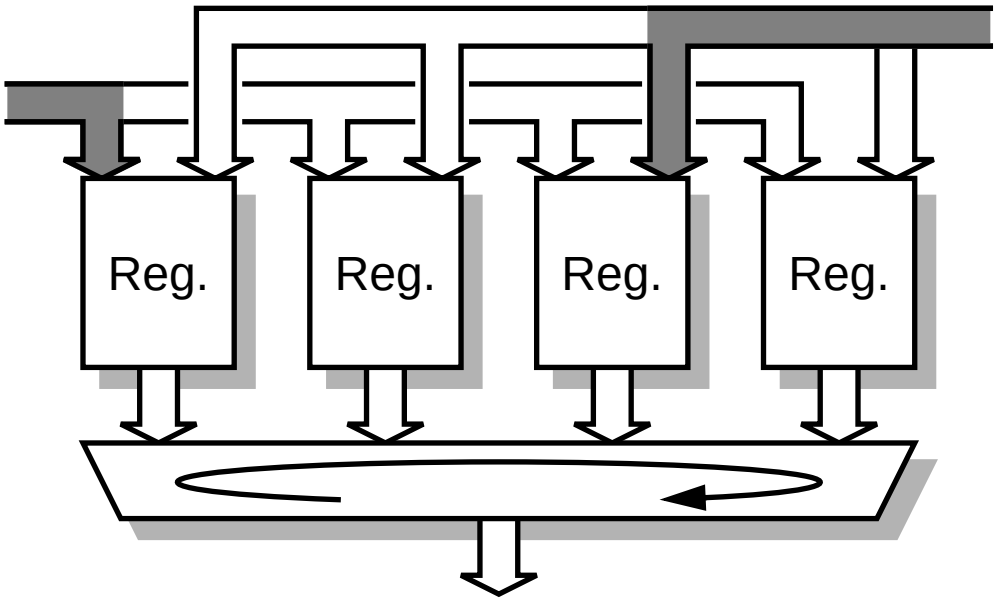


- It allows instructions to complete in any order.
- It resolves register dependencies.
- It allows register forwarding.
- It permits low-overhead memory management
- It supports exact page fault exceptions

All of this and asynchronous too!

Reorder Buffer

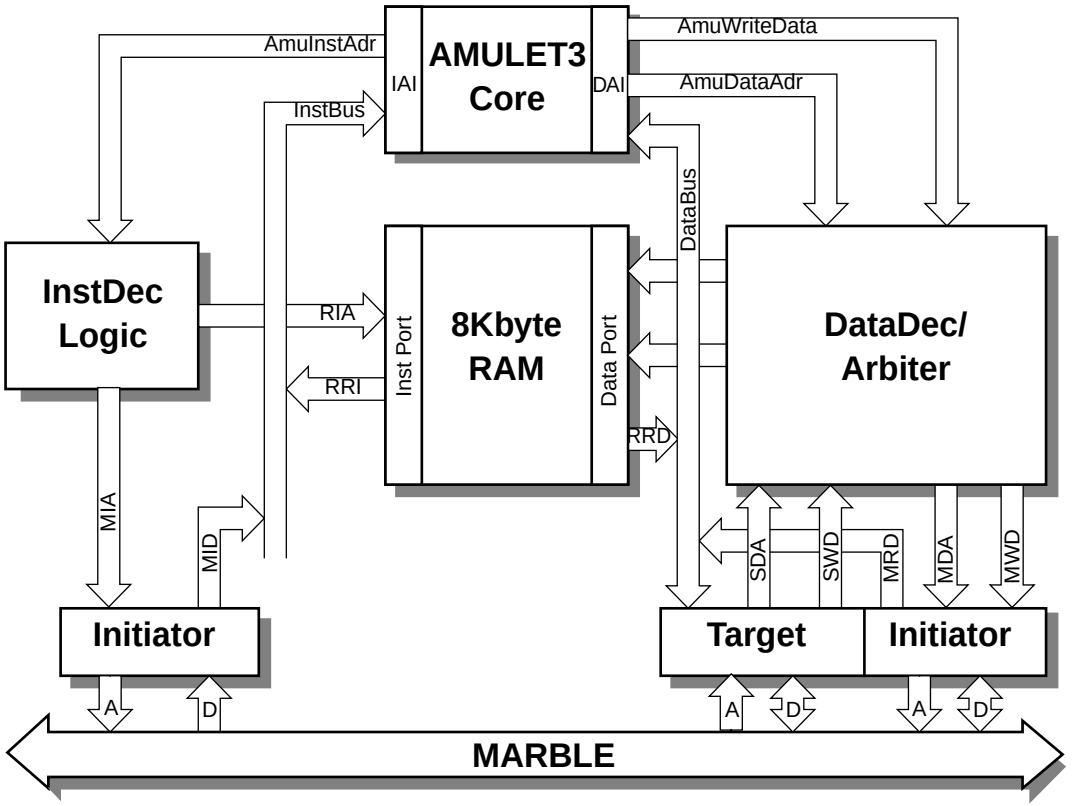
The insight is obvious – (with hindsight):



- Data can arrive down any path at any time, providing their targets are mutually exclusive
- Read out waits for each register to be filled in turn, then copies out the result (or not if unwanted)
- Copy out frees the register but *does not delete the data*

Memory system

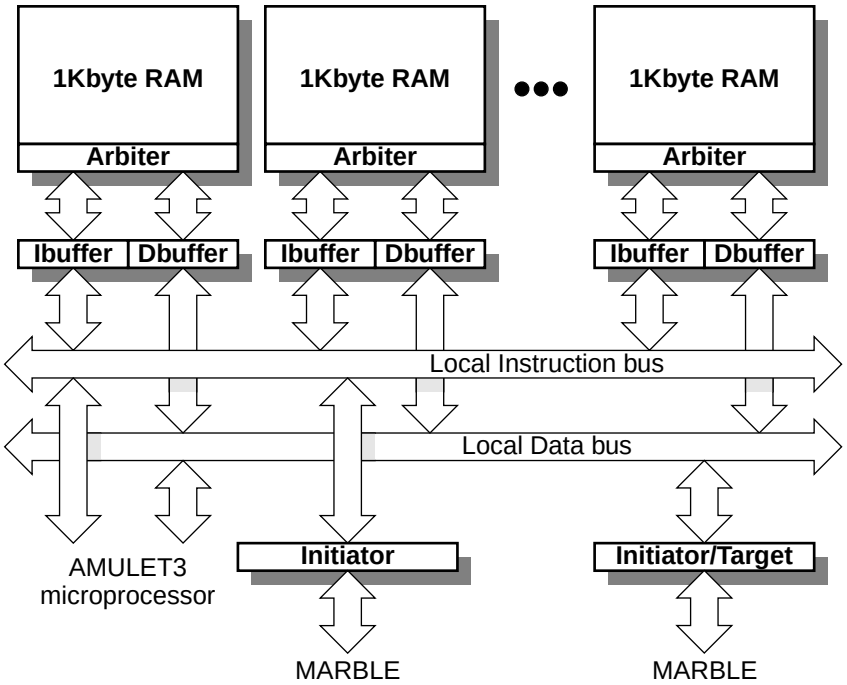
8Kbytes of RAM is accessible via two ‘local’ buses



- The RAM is ‘dual-port’ (at this level)
- The instruction bus is simpler so it has a higher bandwidth

Memory structure

The local RAM is divided into 1Kbyte sub-blocks



- Unified RAM model
- Close to dual-port efficiency

Roughly half of instruction fetches are satisfied from the ‘Ibuffers’

MARBLE

- Centrally arbitrated, multi-channel, asynchronous on-chip bus
- Separate, decoupled transfer phases for address and data
- Standard ‘master’ and ‘slave’ interfaces

Supports: 8-, 16- and 32-bit transfers, bus locking, sequential bursts, …

Synchronous bridge

- A slave interface for clocked peripherals
- Performs synchronisation in the usual way (with usual risks)

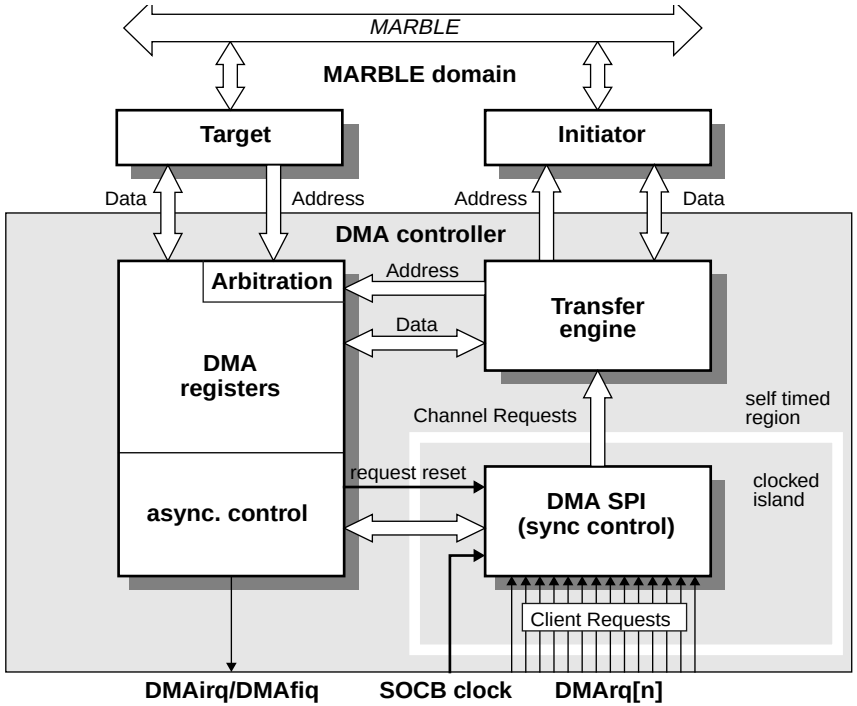
Supports: conventional clocked peripherals.

External bus interface

- Self-timed memory interface (software calibratable delays)
- Usable as external test interface

Supports: 8-, 16- and 32-bit memories, SRAM, DRAM, …

DMAC – making models with Balsa



- About 70 000 transistors
- Regular structures (i.e. register banks) in full custom design
- Control synthesised from Balsa description (first sizable example)
- Cheats slightly by letting a clock into one corner

AMULET3i – Vital Statistics

Transistor count

- AMULET3 – 113 000
- RAM (total) – 504 000
- DMA controller – 70 000
- EMI – 26 000
- Total – 800 000 (asynchronous subsystem)

Geometry

- 0.35µm, 3 layer metal (using ARM's generic design rules)

Area

- AMULET3i – ~25mm^2
- AMULET3 – ~3mm^2

Note: the local RAMs are relatively large in these generic, ASIC rules.

System Performance (Measured)

- Peak Native MIPS 79 MIPS (96 in Thumb code)
- 149 kDhrystones[1]/s – 85 Dhrystone MIPS (ARM)
- 108 kDhrystones/s – 62 Dhrystone MIPS (Thumb) (-30%)

- AMULET3i power average 130 mW
  - 60% is within the processor core (simulation result)
- 660 MIPS/W for the system
  - 1100 MIPS/W for the processor core

About 15% slower than expected – awaiting silicon process information

For comparison:

- 0.35µm ARM9 ⇒120 MHz, (133 Dhrystone MIPS)
  800 MIPS/W

---

1. Dhrystone 2.1 benchmark (normalised to VAX MIPS)

Bus Speeds (Simulated)

Local RAM bandwidths

Speed depends on bus and whether the ‘level 0 cache’ hits or not.

- Instruction bus ‘hit’ 9.5ns (105Mwords/s)
- Instruction bus ‘miss’ 12ns (83Mwords/s)
- Data bus ‘hit’ 13ns (77Mwords/s)
- Data bus ‘miss’ 16ns (63Mwords/s)

In typical code >50% of instruction fetches are ‘hits’.

MARBLE

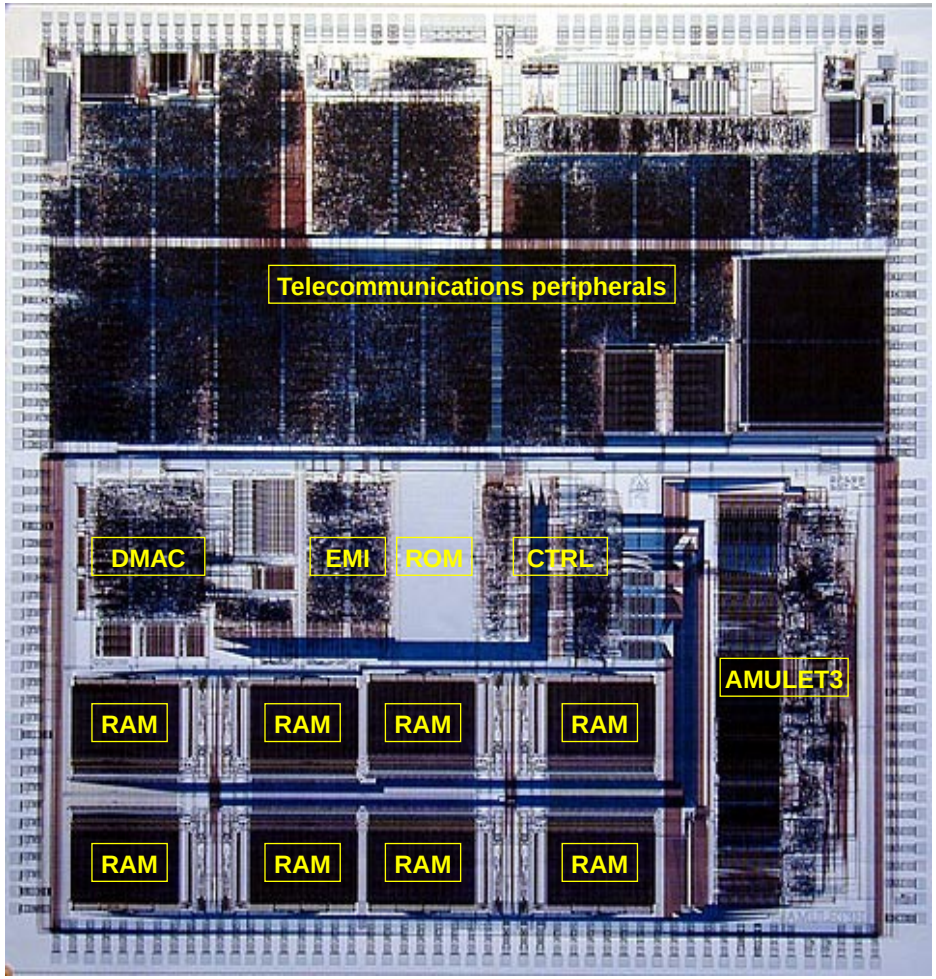
- Total bandwidth – 85Mword/s
- For any one initiator – 55Mwords/s

Caveat: these are from simulations – the absolute numbers may be lower.

Comments

- AMULET3i is about 2x faster than AMULET2e
  - The speed-up is about 1.4x when normalised for the different processes (0.35µm vs. 0.5µm)
  - This is less than was expected
- The performance is heavily limited by memory bandwidth
  - There should be another 30% here (CPU >100 MIPS)
  - (The designer moved continents too soon!)
- The Thumb decompression logic is the limiting factor in Thumb code
  - Speed was not a design priority here
- Simulated performance not met (not yet known why)
  - MIPS -15%
  - MIPS/W +35% – considerably better than ARM9!

DRACO



DECT Radio Communications Controller

Conclusions

Asynchronous logic:

- can be competitive with ‘conventional’ designs
- has particular advantages with low-power and low EMI
  - think portable systems
- may be the only solution to some tasks on big chips
  - especially block interconnections

but

- designing big systems is a *lot* of work
- it’s hard to catch up with the big companies

www.cs.man.ac.uk/amulet/projects/AMULET3i.html