AMULET2e

Jim Garside

Department of Computer Science, The University of Manchester,

Oxford Road, Manchester, M13 9PL, U.K.

http://www.cs.man.ac.uk/amulet/

jgarside@cs.man.ac.uk

What is it?

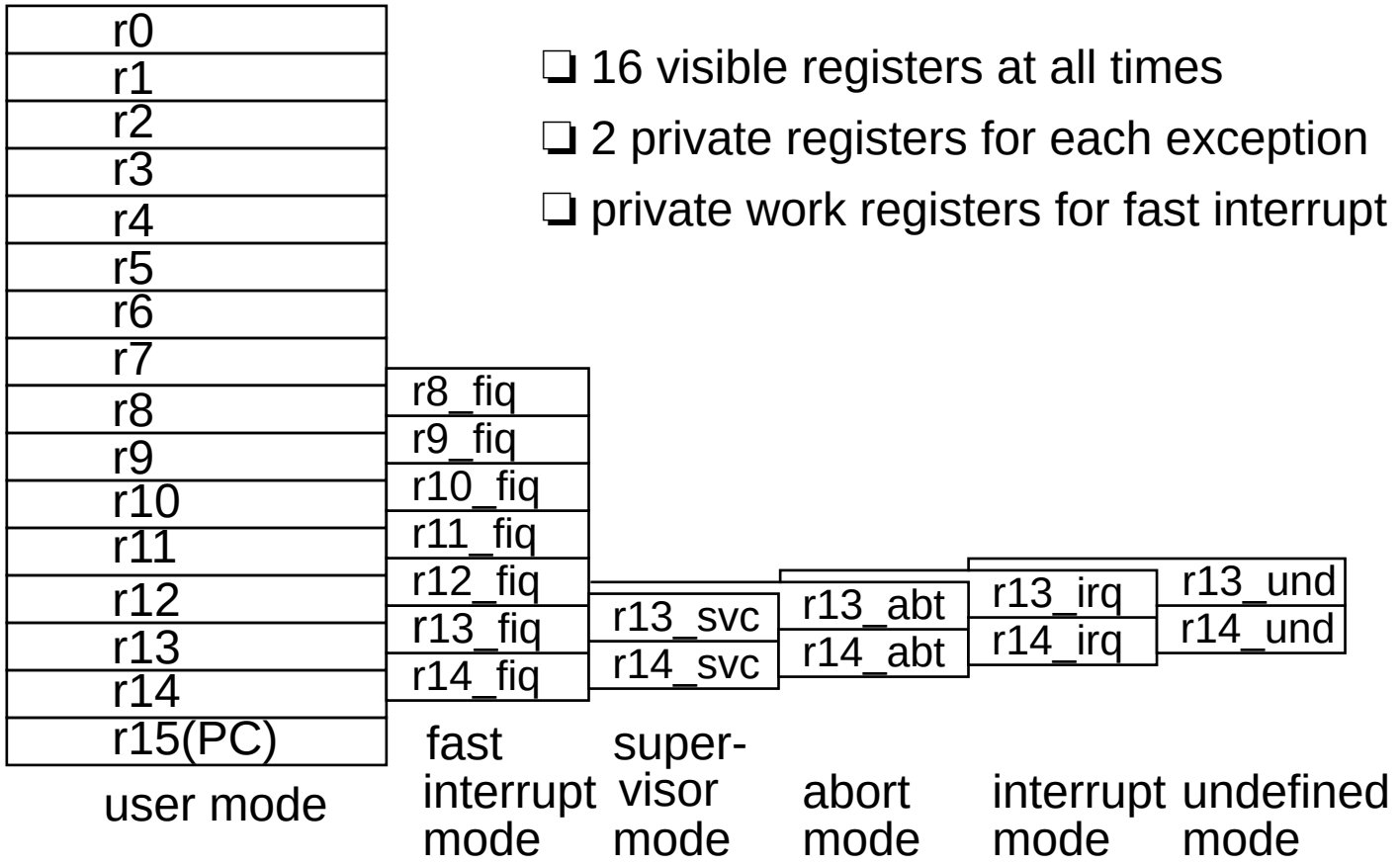
- AMULET2e is an implementation of the ARM architecture
- It is a ‘system’ chip complete with cache, bus interface etc.
- The microprocessor contains performance-enhancing features
- It is completely asynchronous (i.e. self-timed)

Contents

- The ARM processor
- Asynchronous pipeline design
- The AMULET1 and AMULET2 microprocessor cores
  - Some example problems and solutions
- The AMULET2e chip
- Results and analysis
- Conclusions

The ARM processor

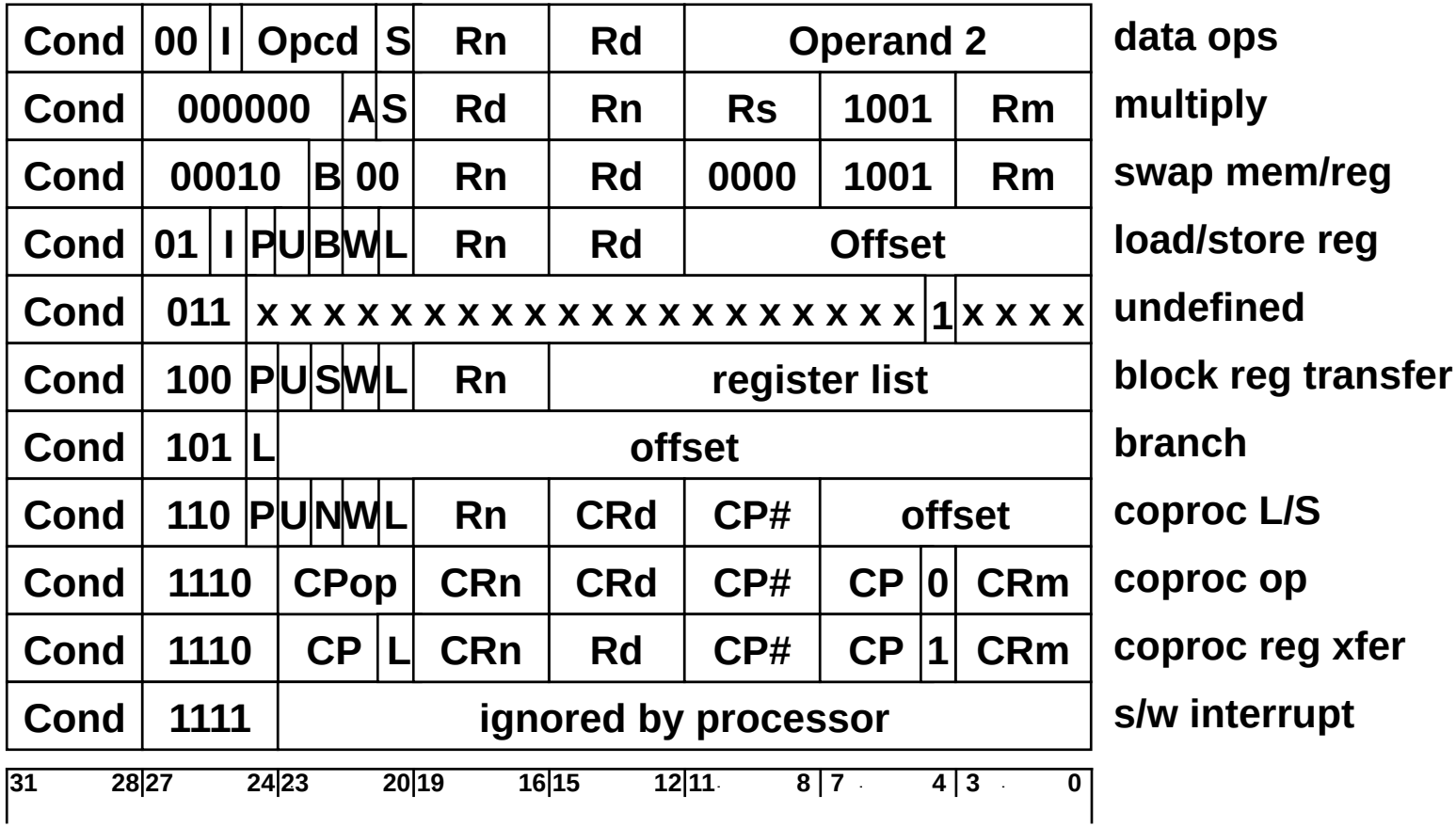
General registers and Program Counter



- 16 visible registers at all times
- 2 private registers for each exception
- private work registers for fast interrupt

The ARM processor

The ARM6 instruction set



ARM as a demonstrator

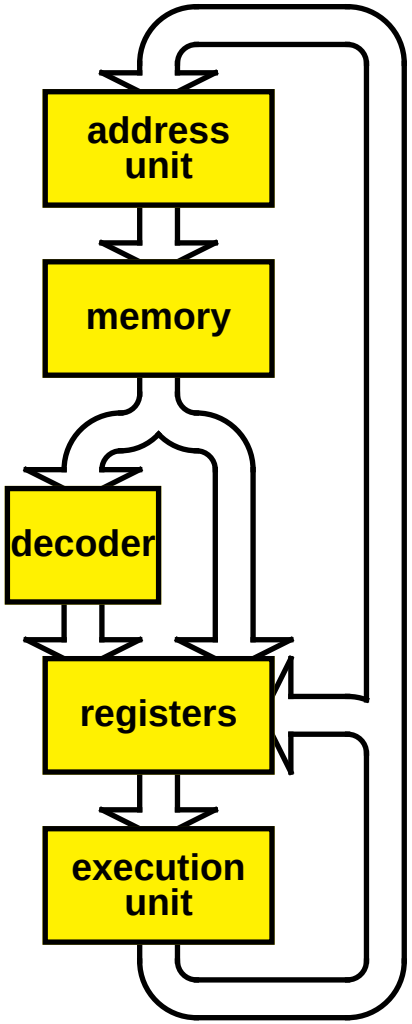
Why choose an ARM as a technology demonstrator?

- ARM is a real, commercial microprocessor architecture
  - realistic demonstrator
  - software available
- Low power operation is the primary motivation
  - ARM is a leading low power architecture
- There are difficult problems to solve
  - some complex instructions (e.g. load multiple)
  - interrupts
  - data aborts

Why Asynchronous?

- Power saving
  - Every logic transition requires some energy transfer
  - Fewer transitions means lower power
  - Don't 'clock' any unit when it is not required
- *Potentially* fast
  - Allows cycle time to be data dependent rather than always 'worst case'
  - Can optimise frequent instructions and allow rare difficult operations more time
- Composability
  - Circuits can be assembled as "plug and play"
- Reduced EMI
  - Radiated noise should be broad-band

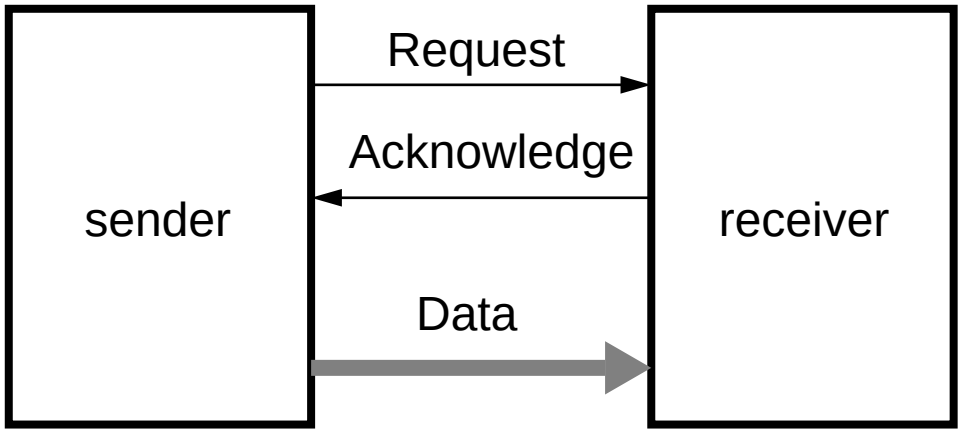
How does it work?

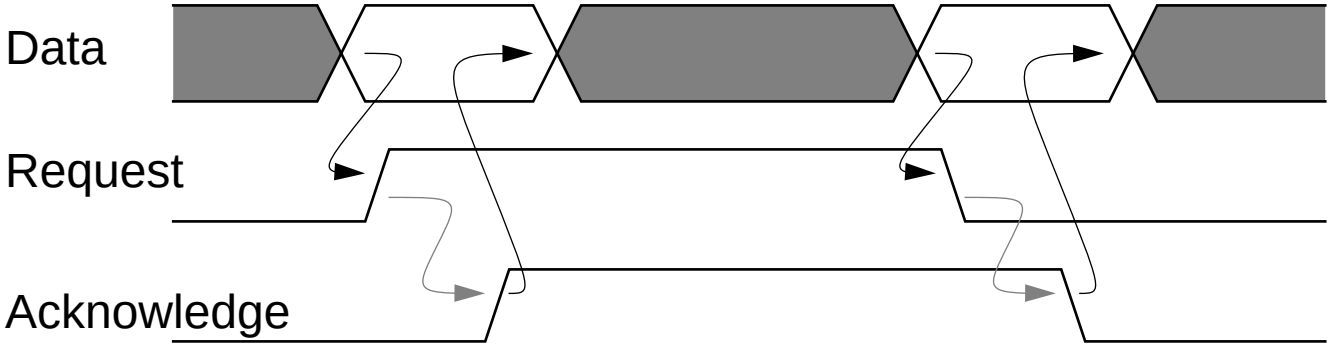


- Each unit runs autonomously, processing at its own speed
- Interactions occur through defined interfaces
- Communication is localised and performed by handshaking
- Timing and pipeline depth is a detail of functional unit design
- Each block can be decomposed similarly into smaller functional units

Inspiration ...

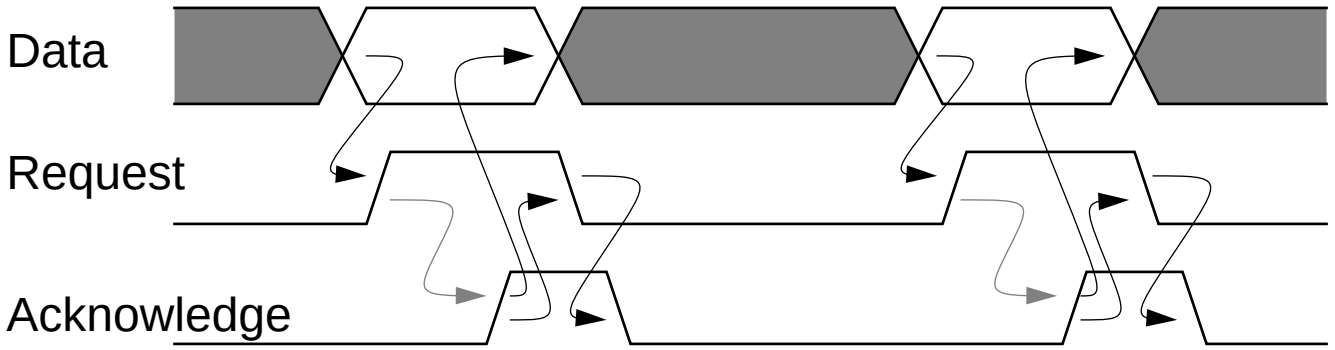
Sutherland's *Micropipelines* (Communications of the ACM, June 1989)





- Bundled data interface with transition signalling

4-phase bundled data protocol

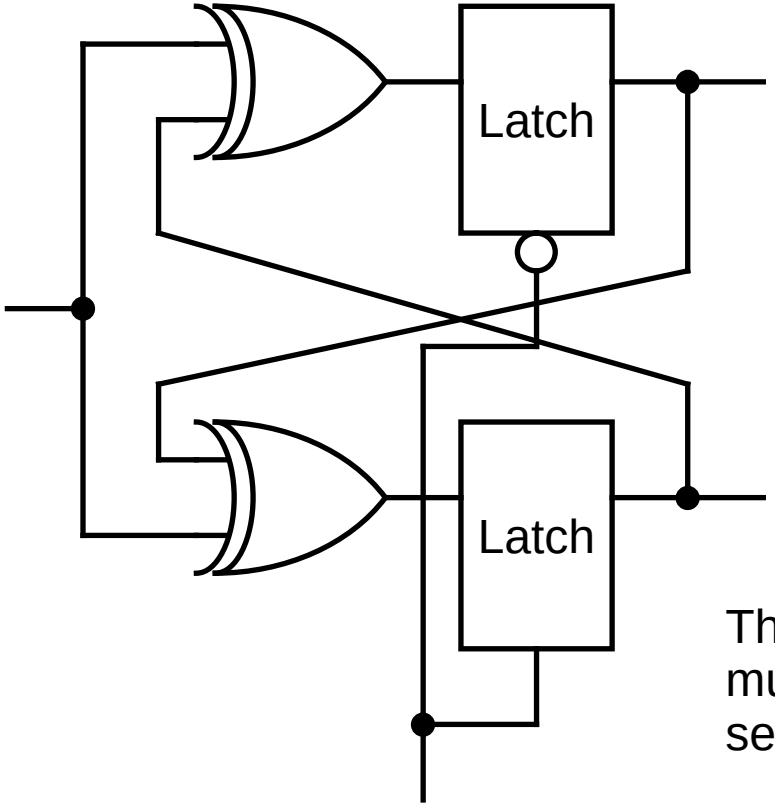


- the choice of active edges is arbitrary
- recovery transitions have no ‘meaning’ and therefore no well-defined timing
- if recovery transitions follow the same path as the active transitions the control cycle time will be long
- various ‘data hold time’ protocols may be employed
- easier to make fast latches and control circuits

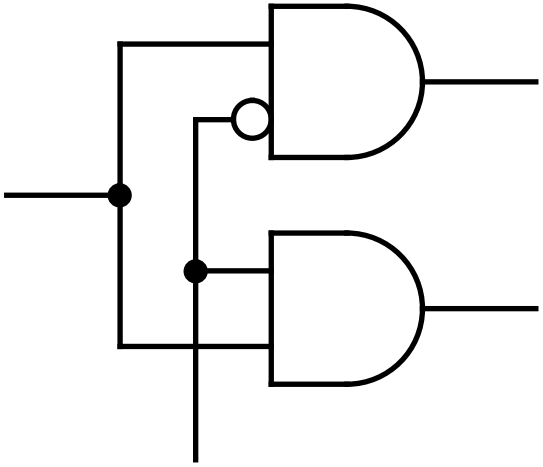
2-phase versus 4-phase

4-phase control circuits are easier to build

2-phase data selector



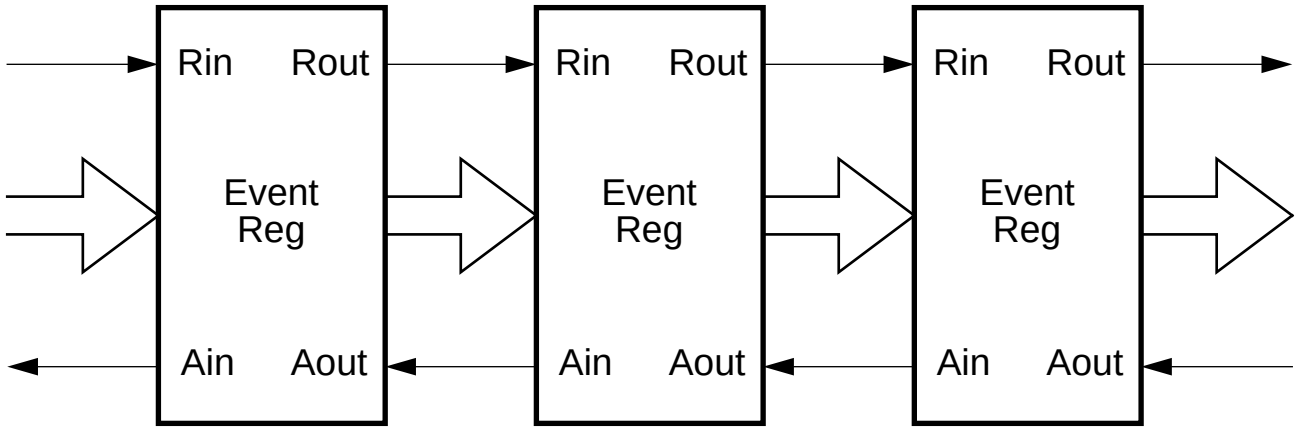
4-phase data selector



The 4-phase also design scales much more easily to ‘wider’ selectors

Micropipelines

A FIFO - the canonical micropipeline:



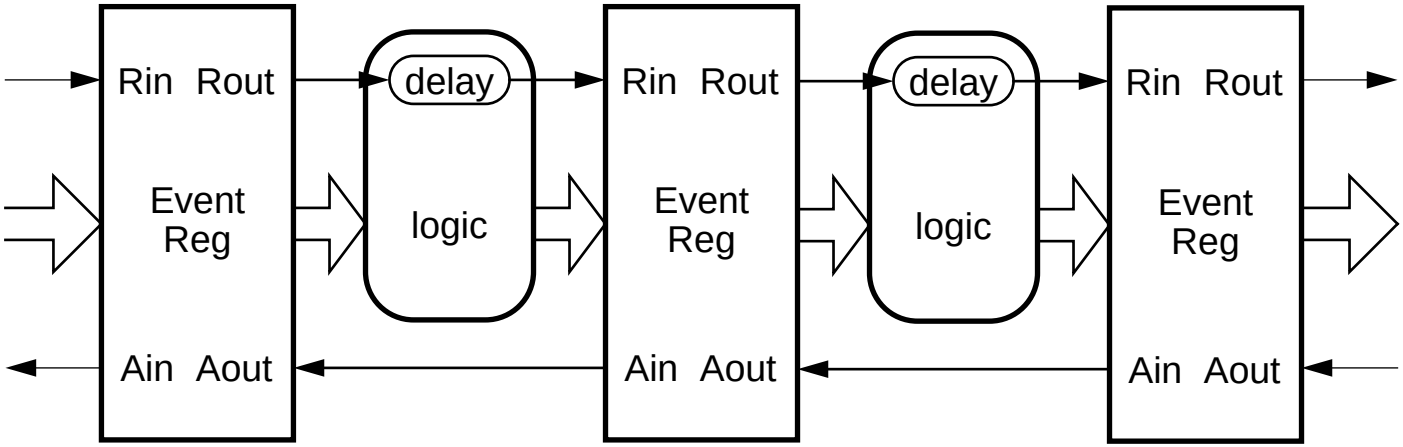
The FIFO has two important performance parameters:

- Latency - how fast will new data pass through an empty FIFO?
- Throughput - what is the maximum sustainable data rate?

These are not as tightly related as they are in a synchronous pipeline.

Micropipelines

A FIFO with processing:



- Processing logic delays the forward data signals
- The “Request” signal should be delayed by the same time
  - Longer delays impact performance
  - Shorter delays reduce functionality!

Delay modelling

Modelling a propagation delay with another propagation delay is potentially hazardous

- Several techniques are used:
  - Registers have a 33rd bit which always makes a transition
  - ALU (adder) has delay routed along the carry chain according to the input data
  - Dynamic circuits (e.g. PLAs) already self-timed
- The delay model is local:
  - Same size transistors with same loads
  - Same manufacturing conditions
  - Same operating conditions (temperature, voltage)

Data dependent delay examples

- Adder delay means that:
  - *typical* additions can be fast without elaborate fast carry logic
  - slower cycles for rare slow cases
  - increment is typically very fast
- Cache cycles faster on sequential cache line accesses
  - CAM look-up averted
  - RAM not precharged between cycles
- Multiplication
  - multiplier uses carry-save adders, local iteration and early termination
  - cycles much faster than other pipeline stages
  - no delay or power consumption when not in use

AMULET1

AMULET1 was the first self-timed implementation of a commercial microprocessor architecture.

- It demonstrated that systems of this complexity were feasible
- It delivered about 67% of the performance of the comparable, synchronous part (ARM6) with a similar power consumption

But

- AMULET1 (1993) was first generation AMULET
- ARM6 (1992) was 4th generation ARM

Room for improvements in:

- Processor's internal architecture
- Circuit implementation

AMULET1 Results

Results of tests on the prototype parts:

	AMULET1/CMP	AMULET1/GPS	ARM6
Process	1µm ES2	0.7µm GPS	1µm
Area	5.5 x 4.1	3.9 x 2.9	4.1 x 2.7
Transistors	58,374	58,374	33,494
Performance	20.5Kdhry.	~40Kdhry.[a]	31Kdhry.
Multiplier	5.3ns/bit	3ns/bit	25ns/bit
Conditions	5V, 20˚C	5V, 20˚C	5V, 20MHz
Power	125mW	N/A[b]	148mW
MIPS/W	77	N/A	120

a. estimated maximum performance
b. the GPS part does not have separate core supplies

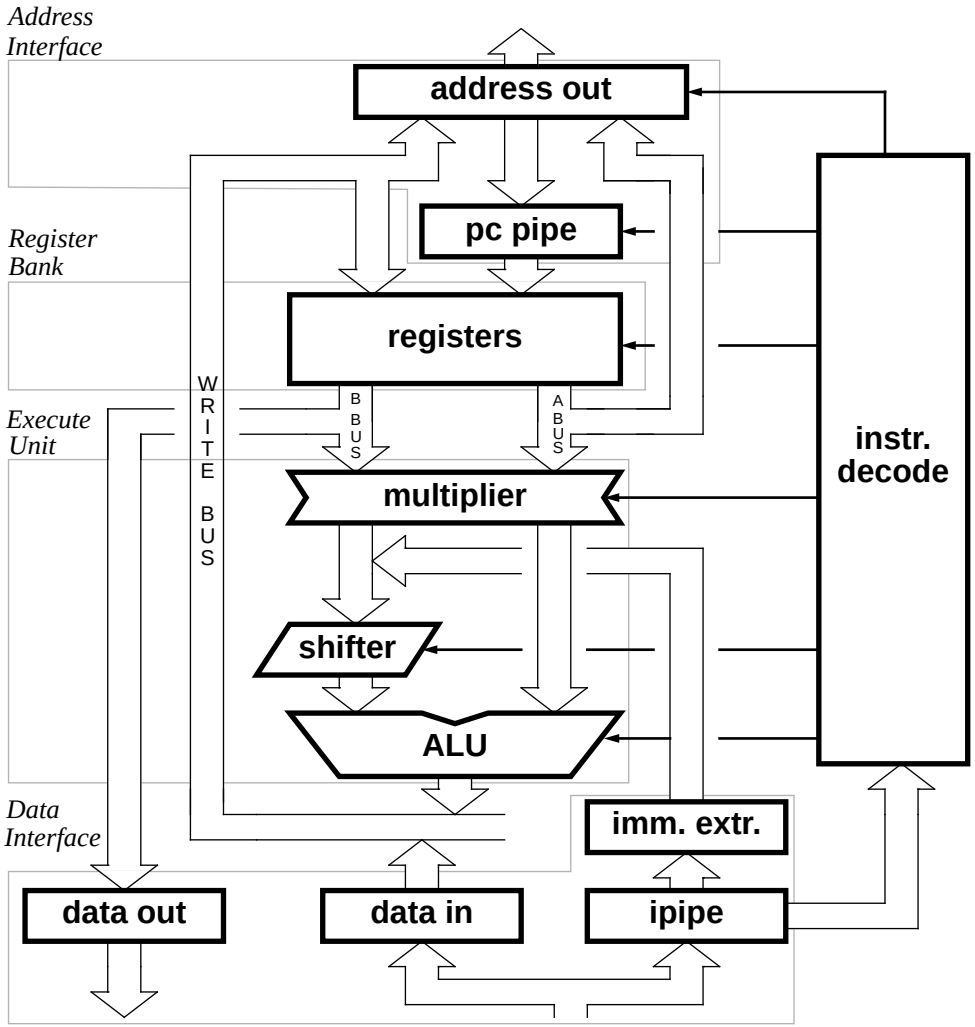
AMULET2

AMULET2 is a the second generation AMULET microprocessor core.

Features:

- Data forwarding
  - load forwarding
  - last result reuse
- Branch target prediction
- “Sleep” mode
  - automatic power-down halt on “loop stop”
- Four phase micropipeline control circuits
  - AMULET1 2-phase control circuits complicated (slowed) latch implementation

AMULET2 Bus Structure



- Conventional bus layout
- Many buses deeply pipelined (not shown)
- Data flow down buses depends only on corresponding units

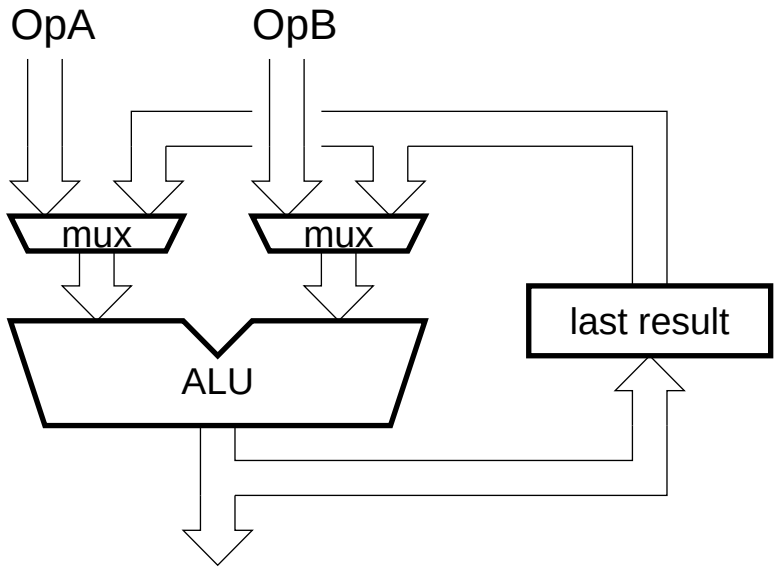
Example: Register Forwarding

Problem: inter-instruction dependencies cause pipeline stalls

Solution: register forwarding

Register forwarding is non-trivial when different pipeline stages are unsynchronised

A localised “last result” register provides a partial solution

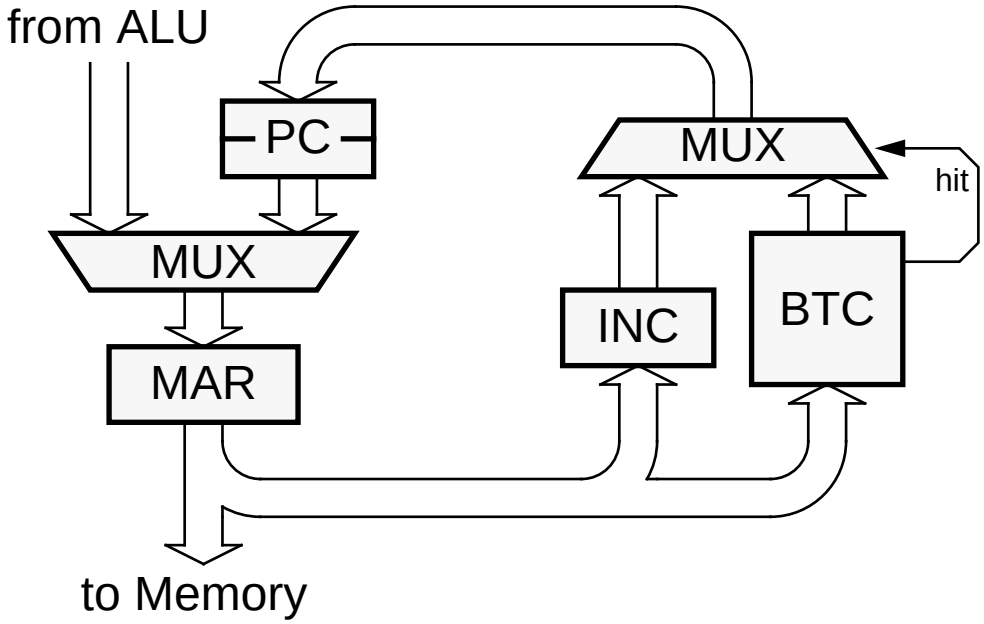


Example: Branch Prediction

Problem: erroneous, speculative fetching wastes time & energy

Solution: branch prediction

- Branch prediction must be performed with purely local knowledge
- A branch target cache can overrule the PC incrementer for invariant branches



Address Interface Operation

- An instruction address is sent from the Memory Address Register (MAR)
- The next instruction address is normally ‘predicted’ by the Incrementer (INC)
- Occasionally an address is recognised as the address of a previously taken branch; the Branch Target Cache (BTC) then predicts a different, non-sequential address

If a non-predicted branch occurs it *interrupts* the loop (asynchronously) and substitutes a new PC value.

- Each sequence of prefetch addresses is “coloured”
- A new PC value changes the prefetch colour
- After a branch, subsequent (speculative) instructions in the same colour are discarded until the new colour arrives
- Prefetch depth is non-deterministic!

Multiplication

Synchronous system

- "Tasks" must occupy an integer number of clock cycles

Asynchronous system

- Periodicity of a given functional unit is arbitrary

Example:

AMULET2 performs multiplication using a free-running carry-save adder unit.

- Cycles in about 30% of the time for a typical instruction cycle
- Performs early termination

A synchronous unit would require either 3x the hardware (to fill the clock cycle) or 3x the clock frequency

AMULET2e

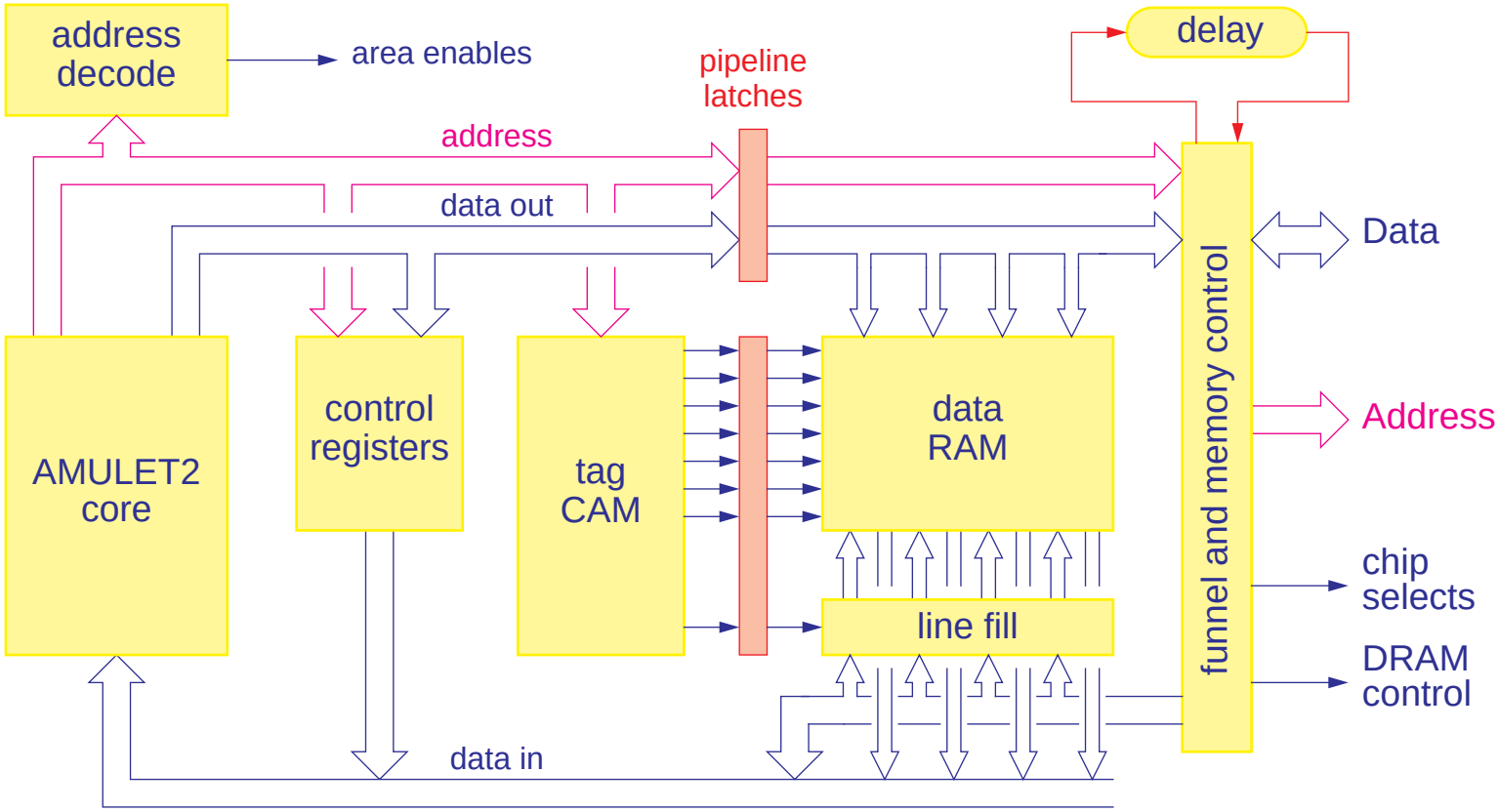
AMULET2e is an integrated processor based around the AMULET2 macrocell.

It comprises:

- AMULET2: a second generation asynchronous ARM
- 4Kbytes of self-timed cache (or memory mapped SRAM)
- a flexibly configurable bus interface
  - supports bus widths of 8/16/32 bits
  - wide range of access times
- The memory interface is simple and designed to interface directly to standard memory and peripheral chips
  - a system can be constructed with few extra components

The processor, cache, and memory controller are all 100% clock-free.

AMULET2e architecture



Cache

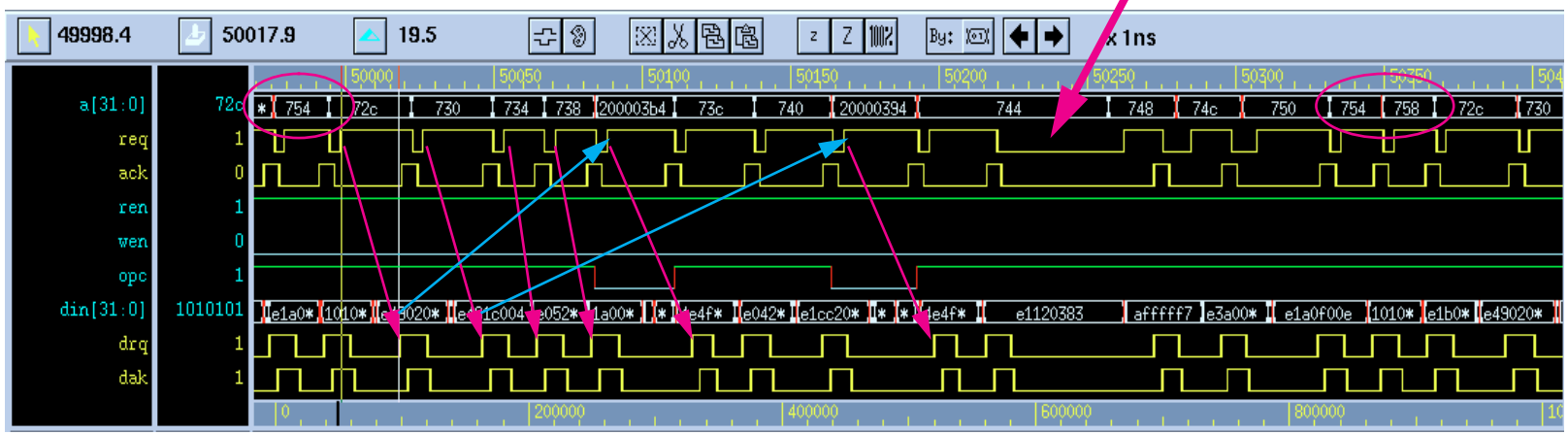
The 4Kbytes of on-board memory are new to AMULET2e

- Comprises 4 independent 1Kbyte cache blocks
  - Each 1Kbyte block is a fully associative, micropipelined CAM-RAM structure
- Cache cycle times vary according to the address pattern
  - CAM look-up is bypassed for sequential addresses in the same CAM line
  - RAM is not precharged between sequential cycles
- The cache line refill process is asynchronous and independent from ‘normal’ cache accesses
  - Forwarding and ‘hit under miss’ are automatic

Example cycle timing

Shows:

- address output (upper 6 traces)
- data input (lower 3 traces)

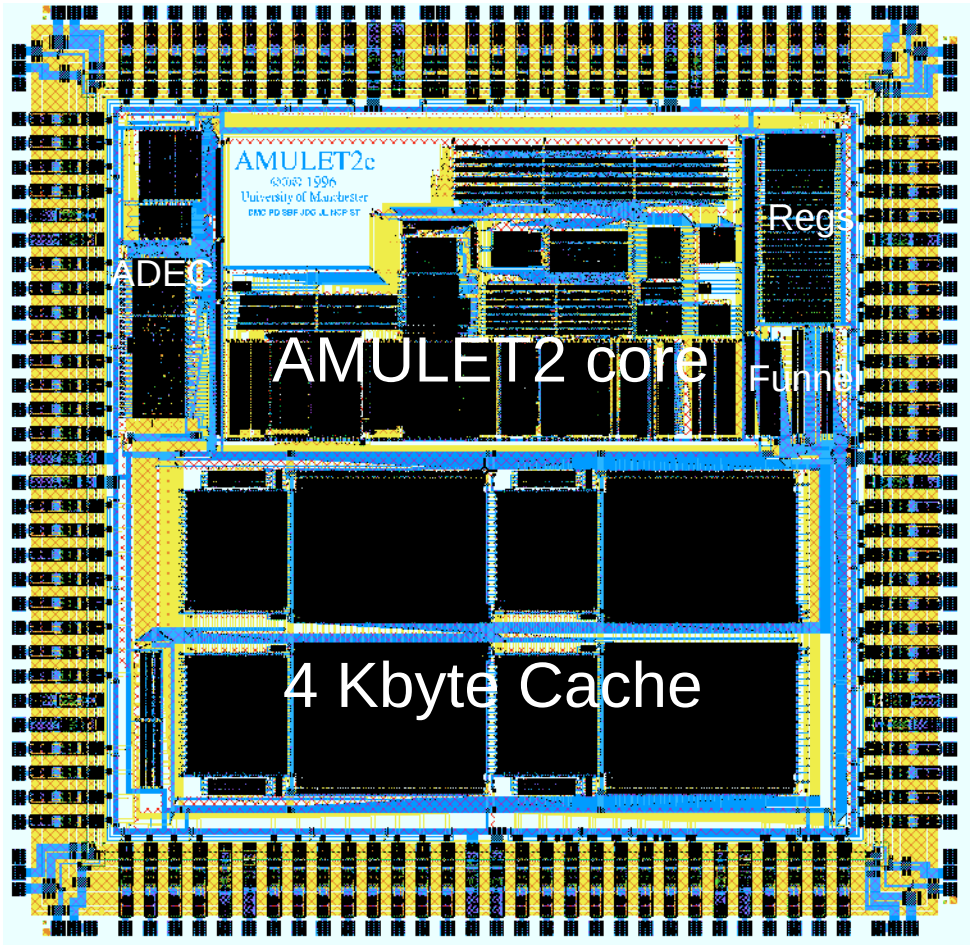



```
72C     LDR     R2, [R0],#+4
730     LDR     R12, [R1],#+4
734     SUBS    R12, R2, R12
738     BNE     &758
73C     SUBS    R12, R2, R3
740     BIC     R2, R12, R2
744     TST     R2, R3, LSL #7
748     BEQ     &72C
74C     ...
```

Note:

- Cycle time varies
- Prefetch depth varies
- Stall on locked register (R12) then prefetch buffer full

AMULET2e layout



Results

- AMULET2 should be a fully functional ARM microprocessor
  - Simulation of layout runs standard ARM code
- 454 000 transistors
  - 93 000 transistors in microprocessor core
- Fabricated on VLSI Technology “cmn5” process
  - 0.5 μm, 3 layer metal process
- Core size 5mm x 5mm; die size 6.5mm x 6.5mm

- Average cycle time around 24ns (TimeMill simulation)
  - ‘Typical’ silicon
  - Room temperature
  - (Cycle time without cache CAM look-up ~ 20ns)
- Faster than AMULET1
  - roughly 30% faster cycling (allowing for process shrink)
  - load and result forwarding reduces stalls
  - branch prediction removes redundant cycles
- Not as fast as ARM8 or StrongARM
- Observations of traces show some unpredicted behaviour
  - there is significant scope for performance improvement

AMULET2e Results

	ARM710	AMULET2e	ARM810
Process	0.6µm	0.5µm	0.5µm
Area (mm^2)	34	42[a]	40
Transistors	570 295	454 000	832 776
Cache size	8Kbytes	4Kbytes	8Kbytes
Performance[b]	23 MIPS	38 MIPS[c]	80 MIPS
Multiplier	20ns/bit	1.7ns/bit	1.7ns/bit
Conditions	3.3V, 25MHz	3.3V, 20˚C	3.3V, 75MHz
Power	150mW	as yet unknown	500mW
MIPS/W	192	as yet unknown	160

a. Includes pad ring
b. Dhrystone MIPs
c. Simulated performance – 'typical' silicon

- AMULET2e submitted for fabrication 11/7/96

Preliminary Result Analysis

- 3.2 x faster than AMULET1
- Performance is roughly half that contemporary ARM (810)
  - Slight loss of ground :-(
- Performance not yet competitive with synchronous chips
- Power consumption measurements await silicon

Excuses

- Moving target
- Only ~12 man years effort
- Many complex pipeline interactions
  - Interactions between stages probably not fully exploited
  - Better tools for load balancing needed
- Cache speed is the major limitation

Critical paths

In an asynchronous system it is difficult to define a single critical path.

The worst areas are:

- Cache
  - especially CAM look up *and subsequent decision tree*
  - this was our first attempt at cache design
- Address generator
  - too many prefetch addresses buffered
  - (surprisingly) the PC increment loop is a slow cycle
- Data processing typically 20%-25% faster than instructions supplied

Some unsolved problems

- A design may be correct yet non-deterministic
  - design verification is a problem
  - ‘balancing’ performance is difficult
- Production test
  - scan paths through latches without a common clock?
  - non-determinism
- Marketing
  - how to sell computers which run at subtly different (and variable) speeds

Conclusions

- Many high-performance techniques can be incorporated in an asynchronous design
  - AMULET2e is roughly 30% faster cycling than AMULET1 (allowing for process shrink)
  - Architectural features also increase performance
- Performance not yet competitive with synchronous chips
- Power consumption measurements await silicon
- Many complex pipeline interactions
  - Interactions between stages probably not fully exploited
  - Better tools for load balancing needed
- Implementation-dependent instruction sets are a bad idea
  - many problems stem from features derived from original synchronous implementation

References

A number of publications on AMULET1 and some on AMULET2 are available electronically:

- http://www.cs.man.ac.uk/amulet/
- http://maveric0.uwaterloo.ca/amulet/

For example:

The AMULET2e Cache System J.D. Garside, S. Temple, R. Mehra
Proceedings: Async'96 Aizu-Wakamatsu, Japan, March 18-21 1996

*Dynamic Logic in Four-Phase Micropipelines*S.B. Furber and J. Liu
Proceedings: Async'96 Aizu-Wakamatsu, Japan, March 18-21 1996

A wide range of information on other asynchronous logic research is also available on this WWW site.

Acknowledgements

- the CEC (ESPRIT project 6909, OMI/DE-ARM)
- Advanced RISC Machines Limited
- VLSI Technology Limited
- Compass Design Automation Limited
- EPIC Design Technology, Inc.
- The AMULET development team