An Asynchronous Fully Digital DLL for DDR SDRAM Data Recovery

Jim Garside et al.
University of Manchester
Contents

- The problem
- Why ‘asynchronous’
- Some asynchronous bits and pieces
- Dynamic switching and glitches
- Overall system
- Results
- Critique
The Problem

- Want a reliable on-chip delay of $\frac{1}{4}$ clock period
- Strobes arrive ‘unexpectedly’ at an arbitrary phase to the internal clock
- Did have a 2x clock from which the SDRAM clock was derived
Existing solutions

- Previous solutions existed

- Some (potentially) not as good
  - e.g. using a fixed delay chain

- DLLs relied on (self) calibration before operation
  - Not adapting dynamically to (e.g.) temperature changes

- Not readily available!
Why ‘asynchronous’?

… especially when the *system* is so synchronous!

We looked at the delay problem ourselves:

- *in ignorance* without preconceptions
- producing a matched delay seemed like something we knew how to do!
- actual delay control is not synchronous with the system clock – not too scary!

The result was:

- a novel solution
- with some useful properties
Delay line

- Delays not particularly remarkable
- Control adopted from Amulet2e
  - Arbitrarily extensible with three control signals
Phase comparator

Operation:

- If one request is more than delay\(^\prime\) ahead of the other it wins both mutexes. When the other request arrives an appropriate correction pulse results.

- If the requests arrive close to each other (within delay\(^\prime\)) lock is achieved.

The (fixed) delays provide a window to prevent ‘hunting’.
Asynchronous state machine

- Modulo up/down counter converts pulses to rolling code
Switching and glitching

- Can the delay be adjusted (safely) whilst in use?

- SPICE simulation suggests so
  - Plot shows worst glitch discovered

- Considerable ‘filtering’ provided by subsequent circuits
- One delay line is used for feedback to calibrate DLL
- Parallel delay lines are used to delay strobes
- Extra delay line provided for fault tolerance
- Other logic used for test and calibration
Layout

- Hand layout from standard cells
- 130 nm process
- 630 µm × 25 µm
Test results

- Measured from silicon
- Wide window of tolerance
- Locks in the centre

- Clock period (ns)
  - 0
  - 2
  - 4
  - 6
  - 8
  - 10
  - 12

- Stages employed
  - 0
  - 10
  - 20
  - 30
  - 40

- 133 MHz SDRAM 'max.' speed
- 'automatic' DLL setting
- Data recovered
- Data not recovered

'Automatic' SDRAM setting
Potential improvements

Like most designs, by the time it’s made there are some new ideas

- **Locking window**
  - There is a fixed time ‘window’ in the phase comparator
  - This could be made programmable to reduce ‘hunting’ around a lock

- **Power saving**
  - The manufactured DLL is continuously calibrated so a clock runs down one line all the time
  - This is dumb! Physical conditions change *slowly*. Once locked the clock need only be sent ‘occasionally’ to confirm or re-establish a lock.
  - It would be possible to ‘gate off’ edges from the ‘tail’ of the line
    They currently always run the full length
Conclusions

- **The circuit works** (well) in silicon
  - Finds ‘best’ delay for required job
  - Allows continuous calibration

- Fully digital solution: all standard cells
  - (with the addition of a mutex)

- Asynchronous ‘mindset’ resulted in ‘unusual’ design
  - Async. state machine used to solve SDRAM problem

- Several inefficiencies in manufactured circuit
  - There were competing priorities at the time! 😊
Acknowledgements

- Co-authors
  - Steve Furber
  - Steve Temple
  - Dave Clark
  - Luis Plana

- Rest of APT group

- Sponsors
  - EPSRC
  - ARM Ltd.
  - Silistix