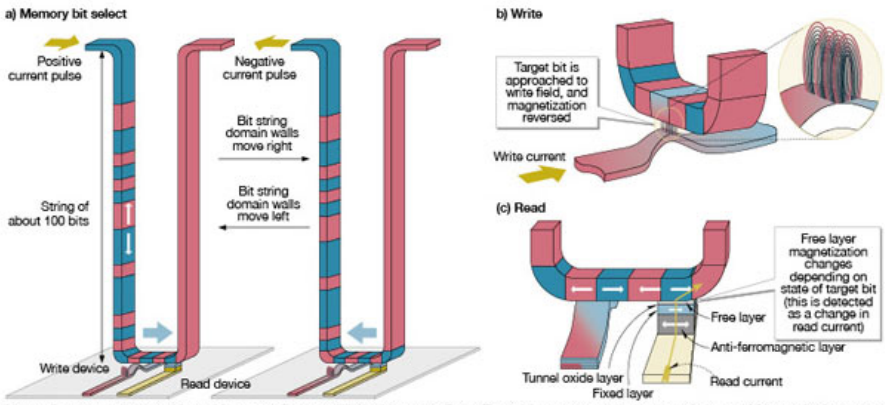
COL719 Project Report

Domain Wall Memory

Introduction

Domain-wall memory, also known as racetrack memory, is an experimental non-volatile memory which is being developed by IBM's Almaden Research Center. A 3-bit version of the memory was demonstrated in early 2008. Racetrack memory has the potential to offer a storage density higher than solid-state memory devices like flash memory.



Using Current Pulses to Move Magnetic Domain Walls Magnetic Race-Track Memory records a string of about 100 bits of information perpendicularly to the Si substrate for each read/write device. This means information density is about 100 times higher than MRAM. Operating principles are shown for access memory bit select (a), write (b) and read (c). When selecting the memory bit for access a current pulse is applied to the magnetic material, causing the magnetic domain wall to move. A positive current pulse will move the wall to the right in the diagram, and a negative one to the left. The quantity of pulses can be controlled for random access, with data read executed after the target bit has been accessed. (Diagram by *Nikkei Electronics* based on material courtesy IBM)

Implications of the Design

As mentioned in the previous section, the design offers a very high storage density. However, this comes at the cost of the memory access times and the access time is not as *random* as in the case of a DRAM or a similar memory design. Instead, the time taken is proportional to the distance between the memory addresses. This happens because the data is shifted to the "reading point" for every memory access.

Thus, compiler optimisation techniques that assume DRAM-like memory will not lead to the best performance if the memory is replace by a racetrack memory and hence we need alternative optimisation techniques to fully exploit the benefits of this memory design.

Proposed Plan of Action

Although the memory designs of DRAM and Domain Wall Memory require different kinds of memory access patterns to reduce the delays, they are still similar in the sense that they both require *spatial locality*. Although the meaning and requirement of this locality differs for both cases, simple modifications and extensions of compiler optimisation techniques can be made to work with Domain Wall Memory (this will avoid the need to reinvent the wheel from scratch).

Apart from the spatial locality optimisations, additional optimisations can be made in the direction of modifying the data storage techniques such as,

1. Having a dedicated cache in conjunction with the memory for storing large arrays which have random accesses - since not all arrays are used at the same time, migrating the array data to a cache (which offers equal latency on average) after reading it sequentially from the racetrack memory might help reduce the delays
2. Modifying the prefetching techniques - since modern processors tend to pre-fetch the data before it is required, this can lead to out-of-order accesses to the memory which might cause back-and-forth data movement in the racetrack memory
3. Explore the interaction between processor cache and memory better - since the memory access patterns matter a lot more than in the case of a DRAM, the interaction between the cache and the racetrack memory needs to be monitored more *strictly* and disallow (or reduce) memory accesses that are too far apart unless necessary