ARM810: Dancing to the Beat of a Different Drum

Guy Larri

90 Fulbourn Road, Cherry Hinton, Cambridge, CB1 4JN, England
Guy.Larri@arm.com
Commercial Drivers For ARM810

• As always, need more Performance
  – Twice the performances of ARM710 on same process

• Embedded, low power, portable applications, requires:
  – Inexpensive = small die, small plastic package, limited pin count
  – Very Focused on Low Power

• Licensing Business Model Requires:
  – Performance increase on commodity processes
  – Low power
  – Portable to commodity 0.6µm/0.5µm 3.3V 3-Layer Metal CMOS processes, migration path to 0.35µm
  – Modular Design: ARM8 Integer Core is separate product
    Cache size variants
    Variant with no MMU
  – Use as an embedded macrocell and stand-alone cached processor.
## ARM810 ...

<table>
<thead>
<tr>
<th>Feature</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Architecture</strong></td>
<td>ARMv4 Processor</td>
</tr>
<tr>
<td><strong>CPU</strong></td>
<td>5 stage pipeline + static branch prediction</td>
</tr>
<tr>
<td><strong>Cache</strong></td>
<td>8kB Unified Cache</td>
</tr>
<tr>
<td></td>
<td>Write-Through and Copy-Back</td>
</tr>
<tr>
<td><strong>TLB</strong></td>
<td>64 entry TLB, 3 Mapping sizes</td>
</tr>
<tr>
<td><strong>Performance</strong></td>
<td>84 Dhrystone MIPs @ 72MHz</td>
</tr>
<tr>
<td><strong>Power</strong></td>
<td>0.5W @ 3.3V</td>
</tr>
<tr>
<td><strong>Die Size</strong></td>
<td>53.5mm² (not including pad ring)</td>
</tr>
<tr>
<td><strong>First Process</strong></td>
<td>CMOS 3-layer metal, 0.6µm drawn</td>
</tr>
<tr>
<td></td>
<td>0.5µm drawn gates</td>
</tr>
<tr>
<td><strong>Portable to</strong></td>
<td>Commodity 0.6µm/0.5µm/0.35µm CMOS</td>
</tr>
<tr>
<td><strong>Package</strong></td>
<td>144 TQFP</td>
</tr>
<tr>
<td><strong>Nice features</strong></td>
<td>Integrated PLL, Lockable Cache/TLB</td>
</tr>
<tr>
<td><strong>Markets</strong></td>
<td>PDA, Network Computer</td>
</tr>
</tbody>
</table>
Pipeline changes for ARM8

ARM 7 Pipeline

<table>
<thead>
<tr>
<th>FETCH</th>
<th>DECODE</th>
<th>EXECUTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction Fetch</td>
<td>Reg Decode</td>
<td>Reg Select</td>
</tr>
<tr>
<td>Reg Read</td>
<td>Shift</td>
<td>ALU</td>
</tr>
</tbody>
</table>

ARM 8 Pipeline

<table>
<thead>
<tr>
<th>FETCH</th>
<th>DECODE</th>
<th>EXECUTE</th>
<th>MEMORY</th>
<th>WRITE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction Fetch</td>
<td>Reg Decode + Select</td>
<td>Reg Read</td>
<td>Complex Shift</td>
<td>Memory Access</td>
</tr>
<tr>
<td>Reg Read</td>
<td>Simple Shift + ALU</td>
<td>ALU Write</td>
<td>ALU Write</td>
<td>ALU Write</td>
</tr>
</tbody>
</table>
**Integer Core CPI Improvement**

ARM8 CPI ~ 1.4  
ARM7 CPI ~ 1.9  

Improvement achieved by

<table>
<thead>
<tr>
<th>Feature</th>
<th>% Improvement over ARM7</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single Cycle LDR</td>
<td>~ 12 %</td>
</tr>
<tr>
<td>Single Cycle STR</td>
<td>~ 4 %</td>
</tr>
<tr>
<td>Double bandwidth LDM</td>
<td>~ 6 %</td>
</tr>
<tr>
<td>Static Branch Prediction</td>
<td>~ 10 %</td>
</tr>
<tr>
<td>Total</td>
<td>~ 35 %</td>
</tr>
</tbody>
</table>
Satisfying Bandwidth Requirement

ARM8 Prefetch Unit.
8 Instruction FIFO & Branch Prediction

Double Bandwidth for Instruction fetch into PU FIFO & for LDM Transfer 32 bits on each clock edge.

Unified I & D Cache

Also gives:
Single memory port for ROM/SRAM (Cache-less) systems.
Small number of Buses
## Cycle Count Summary

<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>Max</th>
<th>Correctly Predicted</th>
<th>Incorrectly or Not Predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Branch</td>
<td>0</td>
<td>3</td>
<td>Correctly Predicted</td>
<td>Incorrectly or Not Predicted</td>
</tr>
<tr>
<td>Branch and Link</td>
<td>1</td>
<td>3</td>
<td>Correctly Predicted</td>
<td>Not Predicted</td>
</tr>
<tr>
<td>Multiply and Accumulate</td>
<td>3</td>
<td>7</td>
<td>$32 \times 8 \rightarrow 32$</td>
<td>$32 \times 32 + 64 \rightarrow 64$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Normal Case</th>
<th>Complex Operand Shift *</th>
<th>3rd read operand</th>
<th>Write to PC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logical</td>
<td>1</td>
<td></td>
<td>+1</td>
<td>+2</td>
</tr>
<tr>
<td>Add, Subtract</td>
<td>1</td>
<td>+1</td>
<td>+2</td>
<td></td>
</tr>
<tr>
<td>Load Word, Half, Byte</td>
<td>1</td>
<td>+1</td>
<td></td>
<td>+4</td>
</tr>
<tr>
<td>Store Word, Half, Byte</td>
<td>1</td>
<td></td>
<td>+1</td>
<td></td>
</tr>
<tr>
<td>Load Multiple Words</td>
<td>$\lceil n/2 \rceil + 1$ where $n = #$ registers</td>
<td></td>
<td></td>
<td>+4</td>
</tr>
<tr>
<td>Store Multiple Words</td>
<td>$n$         where $n = #$ registers</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* = Shift other than LSR by 0, 1, 2 or 3 bits
Cache Features

• 8kB Unified Instruction and Data Cache
• 4 Words per Cache Line, 64-way associative
• Random Replacement Algorithm
• Cache supports Copy-Back and Write-Through operation
  – Selectable per-page in page-table entry
  – Copy-Back reduces write traffic to main memory
    → more main memory bandwidth available for DMA
    → lower system power
  – Write-Though good for frame buffers and easy upgrade from ARM710
• Flexible Cache & TLB locking for real time applications
  – Cache contents can be locked with granularity of 128 bytes
  – Gives low interrupt latency / guaranteed execution time for real-time applications
Cache Implementation

• **Cache is virtually addressed → low power**
  – No address translation required for cache read-hits
  – No address translation required for cache write-hits to copy-back regions

• **Cache stores only virtual tags**
  – Translate addresses for cache-line cast-outs when they occur
  → Avoids storing 512 lines * 25 bits = 1.6kB of physical tags

• **Cache implemented from 1kB CAM-RAM segments**
  – Only 1 segment active for each access
  – Segment selected by 3 bits of Virtual Address
  – Easy to build cache size variants

• **Double bandwidth read port to ARM8**
ARM810 μArchitecture Design Style for Low Power

• **Hierarchy of Clocking Domains**
  – Stop clocks to as much of chip as possible for each stall type or off-chip access.
    e.g. TLB miss stops clocks to Cache and ARM8 until page-table walk completes.
    When TLB requests use of bus, bus controller will stop clock to TLB until
    write buffer empties, making the bus available to the TLB.

• **Separate controllers for each clocking domain.**
  – Yields modular control logic
    e.g. No change to cache control or to ARM8 if MMU is removed.

• **Pipelined Cache and Bus Controller**
  – Maintain 1 write per cycle into cache and write-buffer while giving multiple
    cycles to resolve all controller scenarios.
  – Yields low power via clean synchronous signal transitions on pins.
  – Yields optimal use of sequential bursting on external bus.
ARM810 Clocking Domains

8 Kbyte cache (double-bandwidth)

JTAG
PLL

Write buffer
BIU

address buffer

MMU

CP15

ARM8 integer unit

Prefetch unit

PC Instructions

Virtual Address

Read data
Write data
Copy-back tag
Copy-back data

Update TLB

addressData out
Data in
Data out
32
32
32
physical address

Write buffer

8 Kbyte cache (double-bandwidth)
ARM810 Clocking Domains

Example: TLB Miss following buffered writes
ARM810 Clocking Domains

Wait for write buffer empty

Clocks Stopped
ARM810 Clocking Domains

Page table walk completes

8 Kbyte cache (double-bandwidth)

Copy-back tag

Copy-back data

Virtual Address

read data

Write data

Prefetch unit

PC Instructions

ARM8 integer unit

CP inst

CP data

MMU

JTAG

PLL

Write buffer

BIU

address buffer

Clocks Stopped

Update TLB

Clocks Stopped

8 Kbyte cache

(armbandwidth)

Write buffer

BIU

Address buffer

BIU

Address buffer

PLL
Clock distribution

Family of clock stop elements to drive different loads

G = GLOBAL CLOCK
D = DOMAIN CLOCK STOP
L = LOCAL FUNCTION CLOCK STOP
C = GATED CLOCK
Design Methodology

• Instruction Trace Analysis for Performance Evaluation
• Modelling and high level design validation in VHDL
• Logic Synthesis for complex combinatorial control logic
• Schematics for datapath, latch selection, and clocking
• Extensive static and dynamic timing analysis using EPIC tools on extracted transistor netlist.
• Power consumption sanity check using EPIC dynamic simulation.
• SPICE analysis for CAM-RAMs, datapath, FIFOs.
Implementation Technology

• Full custom layout for Datapath, FIFOs, TLB, Cache CAM-RAMs.

• Standard-Cell for Control Logic.

• Combination of hand-routed and auto-routed layout composition.

• Process portable 0.6µm/0.5µm ruleset with proprietary automated conversion to target process, except ...

... Cache CAM-RAM Segments in target process rules.
ARM810 Development Team