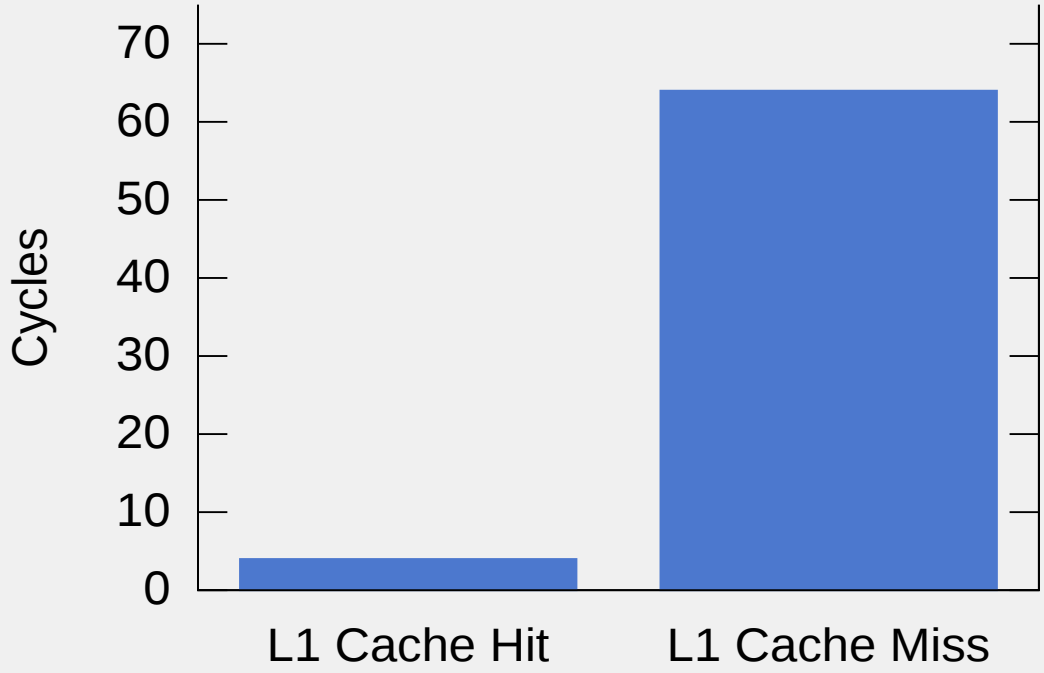
Exploiting On-Chip Memories In Linux Applications

Will Newton, Imagination Technologies

What's wrong with SDRAM?



64 cycles is optimistic

RAM clock often slower than core

SoC fabric and arbiter delays

SDRAM controller bursting delays

TLB miss stalls

It's not just latency

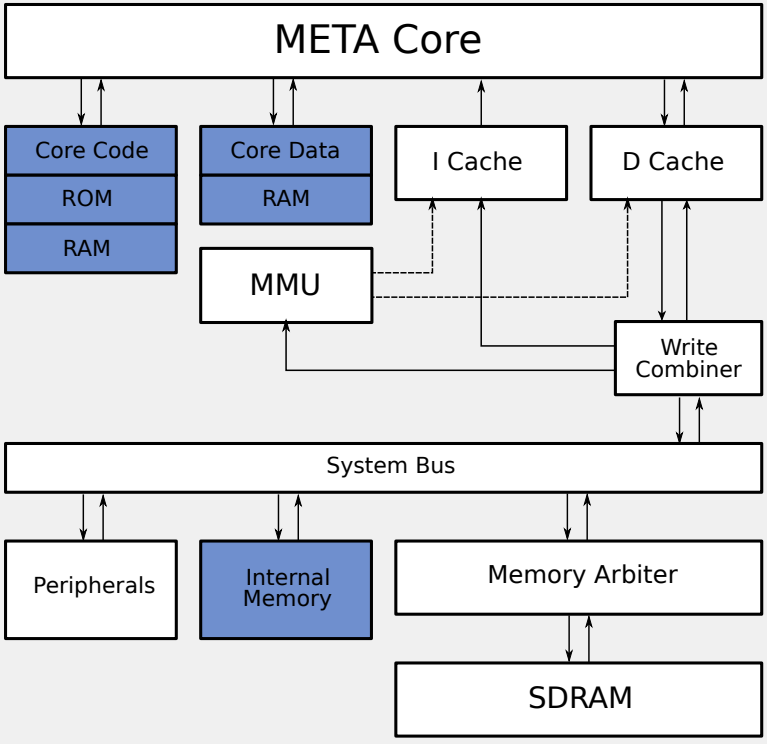
Memory bus bandwidth

Bus contention can affect other cores

Memory bus power consumption

Non-deterministic if you're doing RT

What solutions are available?



Example META SoC

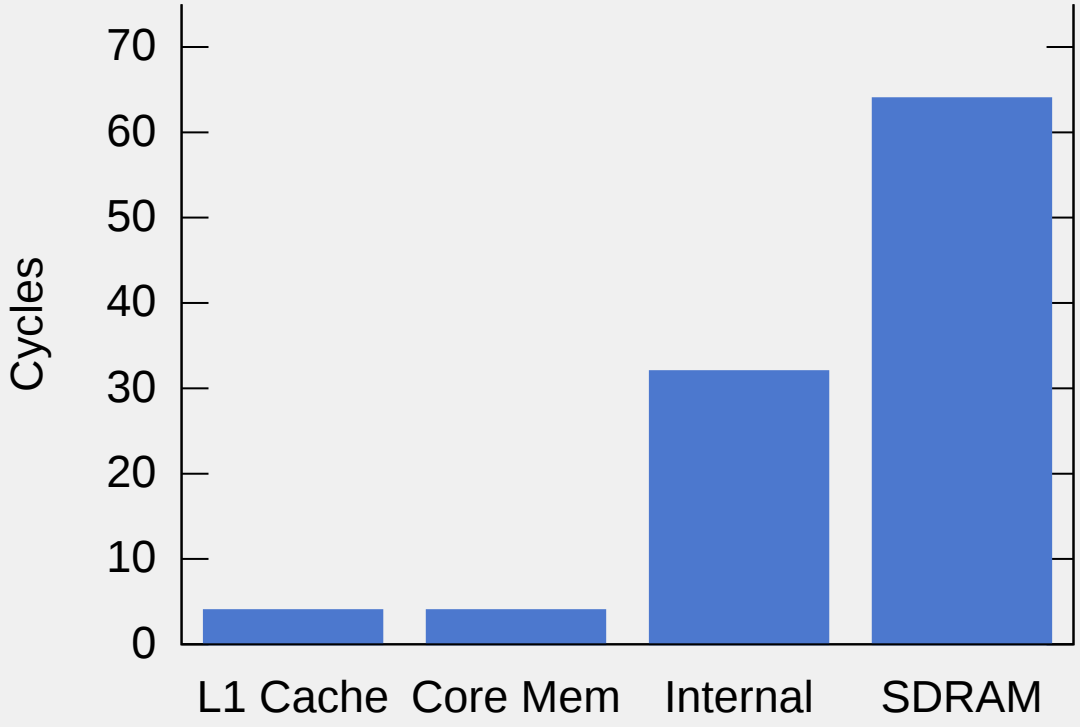
Hardware multi-threaded DSP core

L1 cache - 16k code, 16k data

Core memory - 64k code, 64k data

Internal memory - 384k general purpose

Example META SoC



Using core memories

Ideally we would like usage to be transparent

Fixed addresses make this difficult

Core memory: Executables

Linker script allows placement of sections

```
#define __section(S) __attribute__((__section__(#S)))
#define __core_text __section(.core_text)
#define __core_data __section(.core_data)
static int __core_data mydata;
int __core_text myfunction(int a);
```

elf_map overridden in the kernel

Core memory: Shared libraries

Cannot mix core and MMU in one object

Whole shared object can be placed in core

Only useful for small objects

Core memory: Dynamic allocation

System call API to allocate and free

Can replace specific malloc/free calls

Allows kernel to reserve areas

Core memory: In practice

Not easy to get big speedups

Cache manages small, frequently accessed items well

Beware long branches

Improved tremor decode speed by 11%

Using internal memory

Linux supports cpu-less NUMA nodes

numactl

set_mempolicy(2)

mbind(2)

Internal memory: numactl

Tool to set NUMA policy of an application

```
numactl --preferred=1 ls
```

Does not build easily with uClibc

Too coarse-grained for many situations

Internal memory: set_mempolicy(2)

Sets the memory policy of the current process

```
int set_mempolicy(int mode,
                  unsigned long *nodemask,
                  unsigned long maxnode)
```

Does not move existing pages

Memory policy can be set multiple times

Internal memory: mbind(2)

Sets the memory policy for an address range

```
int mbind(void *addr, unsigned long len, int mode,
          unsigned long *nodemask,
          unsigned long maxnode, unsigned flags)
```

Overrides policy set by set_mempolicy(2)

Capable of moving pages between nodes

Internal memory: In practice

No nice way to implement malloc_from_node(2)

Moving pages can be costly, mbind(2) should be used with precision

Improved tremor decode speed by 8%

Finding hotspots

Code profiling (gprof, oprofile, perf)

Cache profiling (oprofile, perf)

Emulator

Simulator

Where's the code?

Source code for released products

`http://www.pure.com/gpl`

Questions?

will.newton@imgtec.com