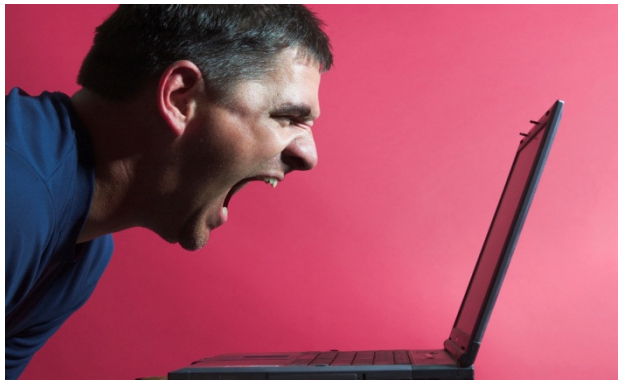
LAUTERBACH
DEVELOPMENT TOOLS

Lauterbach Ltd

www.lauterbach.com

“Over 80% of development time is spent :-

- Debugging your own code
- Debugging the integration with other code
- Debugging the overall system”

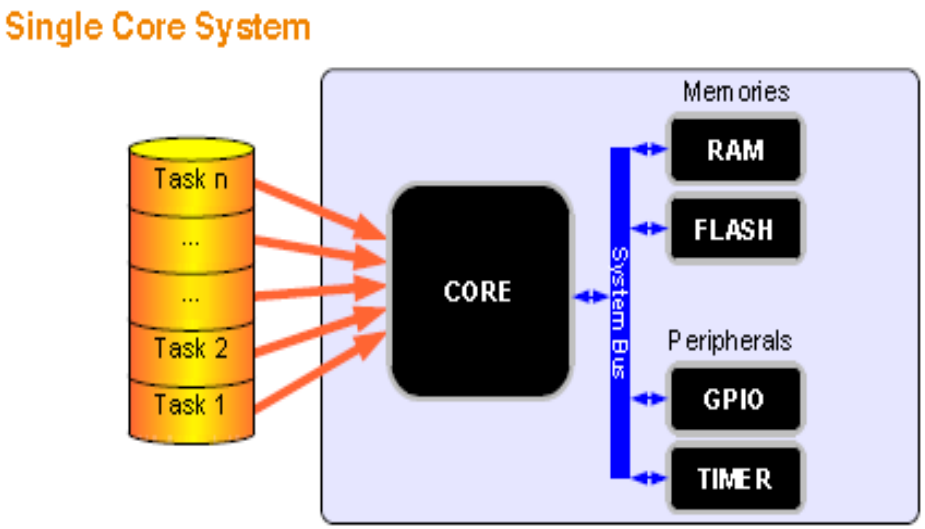


- “It costs 10 to 200 times more to fix a bug at the end of the design cycle rather than the beginning”

- Quotes from ten secrets of embedded debugging

On a single core the code runs sequentially

This makes it easy to set breakpoints and debug



But even here 60% of projects are late to market

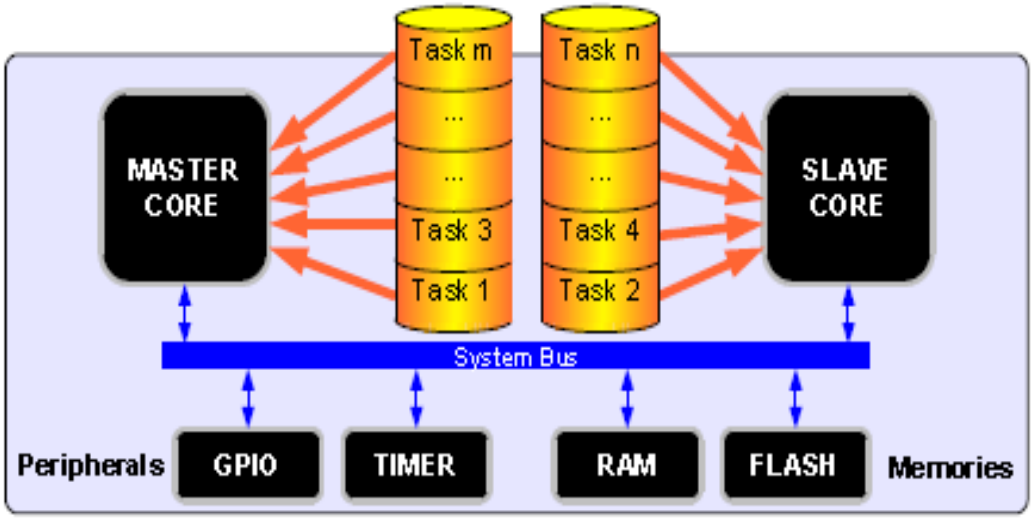
Asymmetric Multiprocessing System (1)

Asymmetric multiprocessing - Dedicated processors for each task
example - your mobile phone which has ARM core(s) and DSP(s)

Code runs independently on each core so most of the code on a core can be debugged as a single entity

The system then needs to be debugged as a whole

Interactions and timing issues can cause problems which are difficult to track.



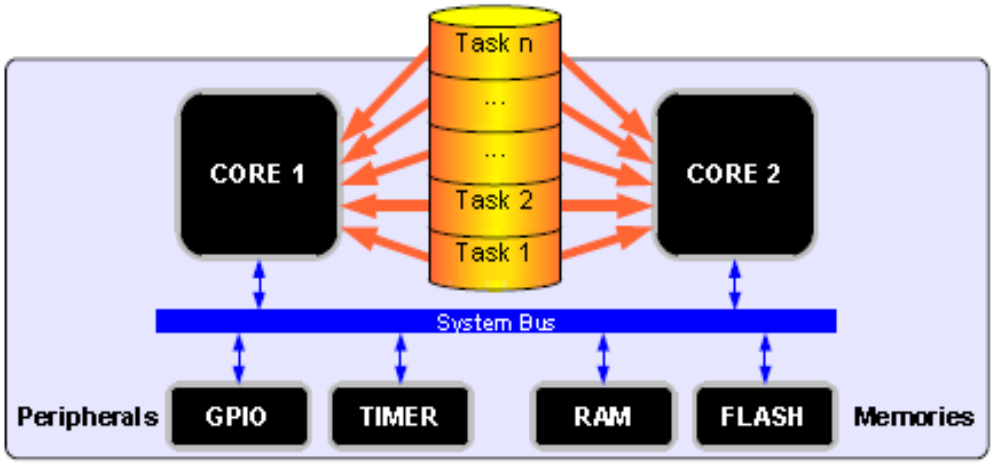
Symmetric multiprocessing

Multiple identical cores
Examples – ARM cores in base stations – Wi-Fi relays etc

Code runs on any core as required

This can make it very complex to debug

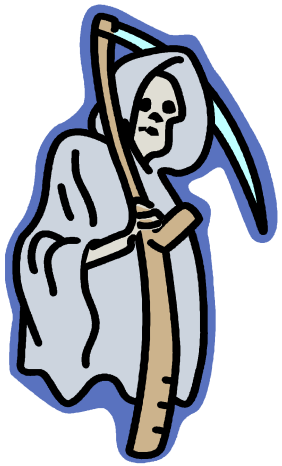
Symmetric Multiprocessing System (SMP) with Equivalent Cores



- Every year in our UK office we see about 20 new or start-up customers. These mostly have small design groups (<10)

- In these customers well over 50% of projects are never completed
- most projects seem to **fail for the same basic reasons!**

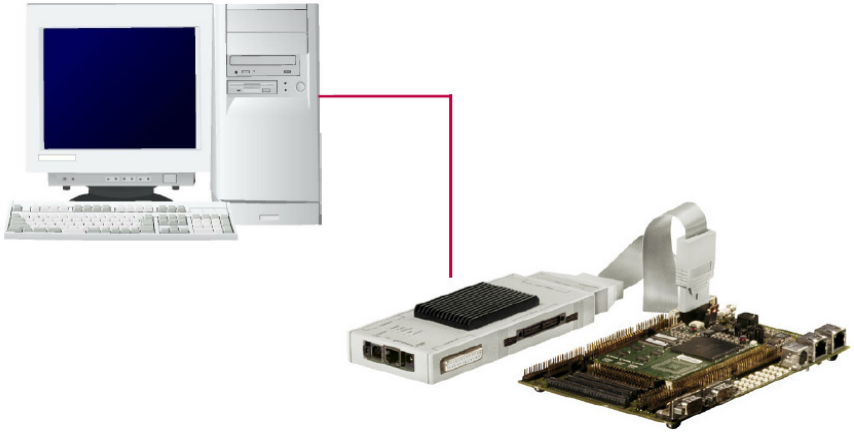
- When you go to **multicore these reasons become even more critical**



- A key item not considered by many design teams is **"how do I debug this system"**

Many factors will interact

- Processor – type – debug ports – trace ports
- Tool chain
- Compiler
- Debug tools



- RTOS
  - Simple
  - Complex with MMU
  - Open source with no support.

The biggest mistake is not thinking “what do we do next”

- An example – choose a **perfect budget core** then decide to put Linux on it . Well the vendor supplied a Linux build for the dev board (and of course Linux is free !).
- The silicon vendor only had to build a bsp that varied slightly from his previous ten development boards
- You will have to build an application that will be many times more complex on a brand new unique board
- To control and debug an RTOS with MMU such as WinCE or Linux you need at least 2 hardware breakpoints –
- To meet your product spec you will need trace to ensure code performance
- BUT you chose a BUDGET core with only one breakpoint and no trace !
- **We currently have 3 customers with this combination of problems**.

Multicore 1

Even more things can go wrong with multicore

We have met a 2 core device designed to run an operating system in SMP mode

But the debug port only has control over core 0

To debug you have to lock any thread into core zero
This effects performance so much that the code operation changes.

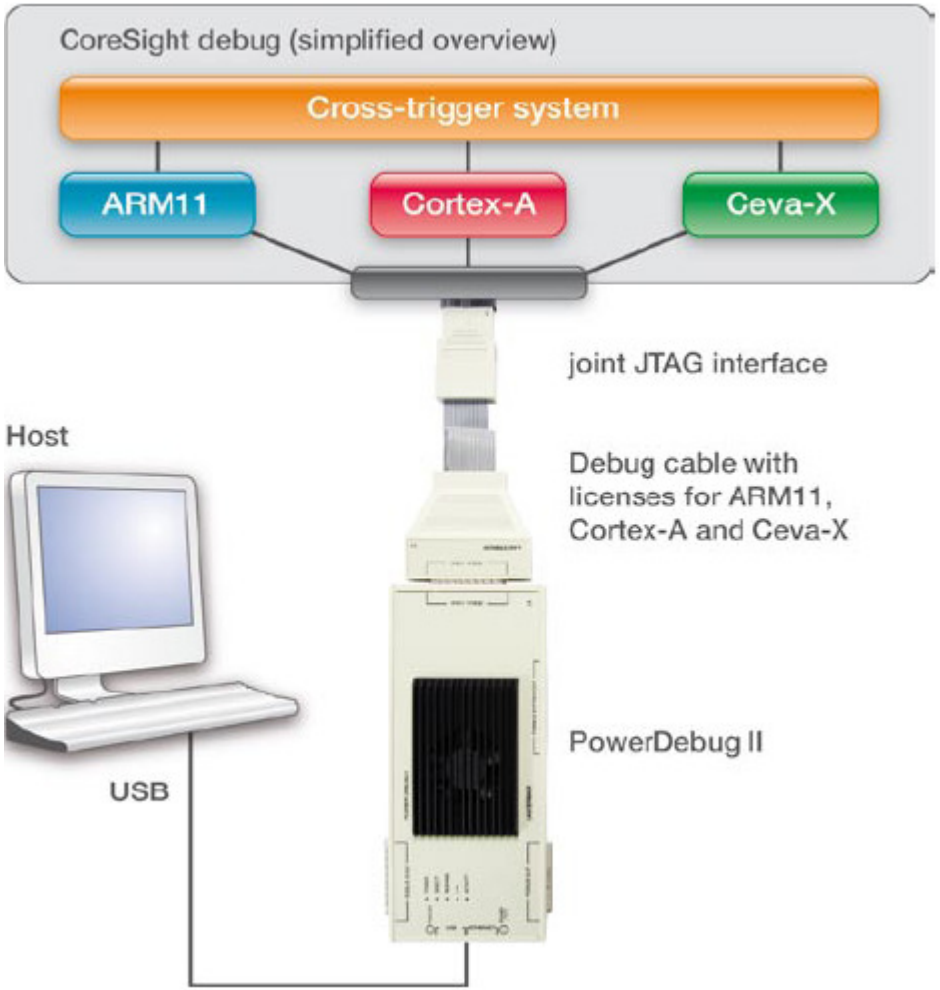
Two of my largest customers have evaluated and decided “ not this chip “

Multicore 2

To control a multicore device you MUST have a good cross trigger matrix

(so that when you stop or break you can control the other cores)

To run in SMP mode you must have hardware breakpoints that will work on ALL cores



Work on your total costs

- Estimate costs of workarounds for problems

(one multicore chip I know has a 360 page errata file)

- Gcc might be free but **we currently see at least 12 variants,** all with their own "quirks" and the end code is normally not as compact as from a commercial compiler.
- If you have to use a larger flash or faster processor Gcc is no longer "zero cost"
- A free rtos is NOT ZERO COST – you will have to do all your own support and bug fixes.
- **Montavista** quote "a dev team of 12 people for 6 months to get a medium complexity Linux build running on a new board".

Choosing a processor

- Some things you might not have thought about in the many hundreds of other decisions you had to take

Choosing a processor 1

- Don't get hung up one feature
- Look at the whole program
- Look at the whole program costs
- There are tens of thousands of microprocessors out there
- Our website lists over 6000 supported types and many of these are just listed by the generic, top version of the family number.
- Last year (2008) there were >2000 NEW ARM cores announced

  (it was also 2000 new ARM cores in 2007)

- You really are spoiled for choice !

Choosing a processor 2

- Number of **hardware** breakpoints
- Mips offers up to 16 as part of the spec – most silicon vendors implement 1 or 2.
- To debug an rtos with an mmu you need 2 or more. With only 1 you are into a nightmare of workarounds just to set a breakpoint.

Essential Multicore features

Hardware breakpoints that can be set on all cores

Cross trigger matrix
(So you can program what cores do in the event of stop/start /step etc)

A poor or non-existent matrix can turn debugging into a nightmare

Choosing a processor 3

Should it have trace ?

- Trace uses more silicon and more pins thus there is an added cost
- Many managers rule it out without thinking further

- If you have an SMP core TRACE is the ONLY way to track interactions as the code runs in parallel on multiple cores

(Note - it must be trace from all cores not just core 1)

Trace - Why spend the extra money?

- Basic Trace Features
  - How the *@!*# did I get here?
  - Which path did it take through here?
  - How long does that take?
  - What was my code doing there?
  - Who's dumping all over there?
  - How often does this run?
  - Which code is actually run?
  - Do my test scenarios cover all possible cases?
  - Can I debug backwards to the cause of a problem?
  - Analyse cache usage and BPU operation

- We have **4 major** TV set top box customers who all need to profile code
- This has been mandated by customers and VARs to clear some performance issues

- **None** of the current dedicated set-top box chips provide the necessary debug info.

So far in the UK since 2005 they have spent over 100 man years trying to sort this problem with limited success !

Impress Your Bean Counters!

- Cost of development
  - Assume 1 year project
  - Assume 5 engineers, 2 hardware and 3 software
  - Each paid £40k per year
  - With overheads cost to company is approx £80K pa. each
  - Total development cost for engineers - £400K pa.
    - Or approx £1500 per day
- Cost of Processor
  - ST Cortex M3 (list prices at 10,000 off Q208)
  - 128M – JTAG and 128K FLASH = US$3.60
  - 256M – JTAG, TRACE, More Peripherals and 256K FLASH = US$4.40
  - Difference = US$8000 (£5500)
  - = LESS THAN **4 DAYS** OF DEVELOPMENT TIME

Impress your Bean Counters!

- Cost of tools
  - Assume 1 trace tool (initial ratio)
    - JTAG tools common for all engineers
  - Approx. £6k extra
  - = THIS IS 4 DAYS OF DEVELOPMENT TIME
  - If trace saves more than 8 days of development time during the year it has recovered the “extra cost”

- A major mobile phone company has put in writing to us that averaged over 200 projects the use of Trace tools cut their debug time by 75%.

- Symbian claim that a **one month delay** in a mobile phone hitting the market results in a loss of profit on that project of US$10Million.

Trace summary

- Trace is not a “cure-all”; it’s just a better way of debugging
  - You are still required to think about things
  - It provides you with more data and more analysis capabilities
  - 60% of the cost of developing a mobile phone is the cost of testing the software

Case study 1

- Major mobile phone manufacturer chooses silicon based upon how good the trace port is
- They swallow extra cost of silicon and extra cost of tools because using trace they save 75% debug & testing time
- On most of their projects they reach the market first with a design that works and meets specifications -- They are No1

Case study 2

•Another mobile phone company tries to do everything on a budget.

•Only some of their designs have trace

•They have problems in the field and have to re-flash approx 2000 phones per week in the UK to fix bugs

•We know this figure is correct as we sell tools to all the repair contractors !!! (2 more orders this week)

- **Case study 3**
  - Another mobile phone company tried to produce a major new concept of phone.
  - Trace was available on the silicon but not brought out on their development boards.
  - The new phone missed a major product launch.
  - The complete new generation of phones missed its key market slot.
  - The UK design centres were closed and 150+ staff made redundant.

WARNING

For lots of projects an RTOS with an MMU is a good idea

BUT for trace to follow the MMU as it allocates virtual memory the trace port MUST provide data trace (as well as the program flow) so the tools have the numerical info to track the new addresses.

Have you evaluated the latest debug technologies?

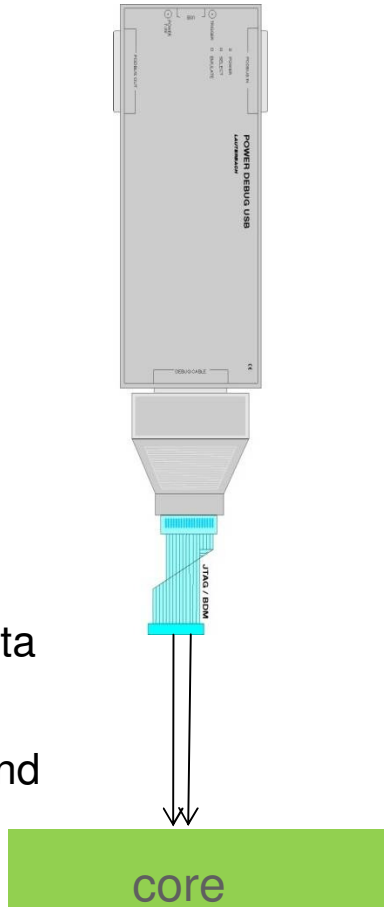
These are not blue sky – they are all in use

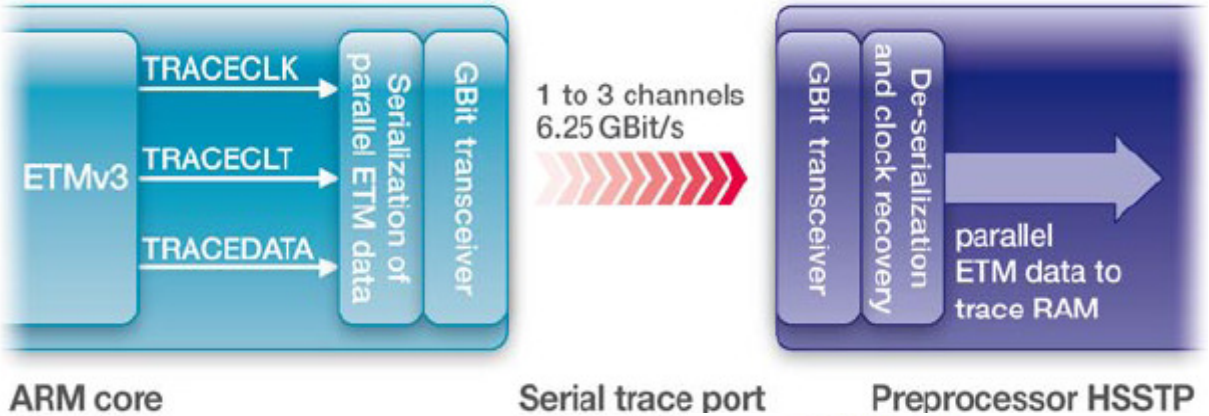
Serial wire debug (JTAG)

- Only 2 pins needed on the core
- Several silicon vendors using this on cores targeted at microcontroller market
- Can provide all the functionality of traditional jtag if implemented correctly in the silicon

- Only clock and data lines needed

(plus Vcc and ground for voltage level reference)





High speed serial trace

(in use since mid 08 on custom silicon)

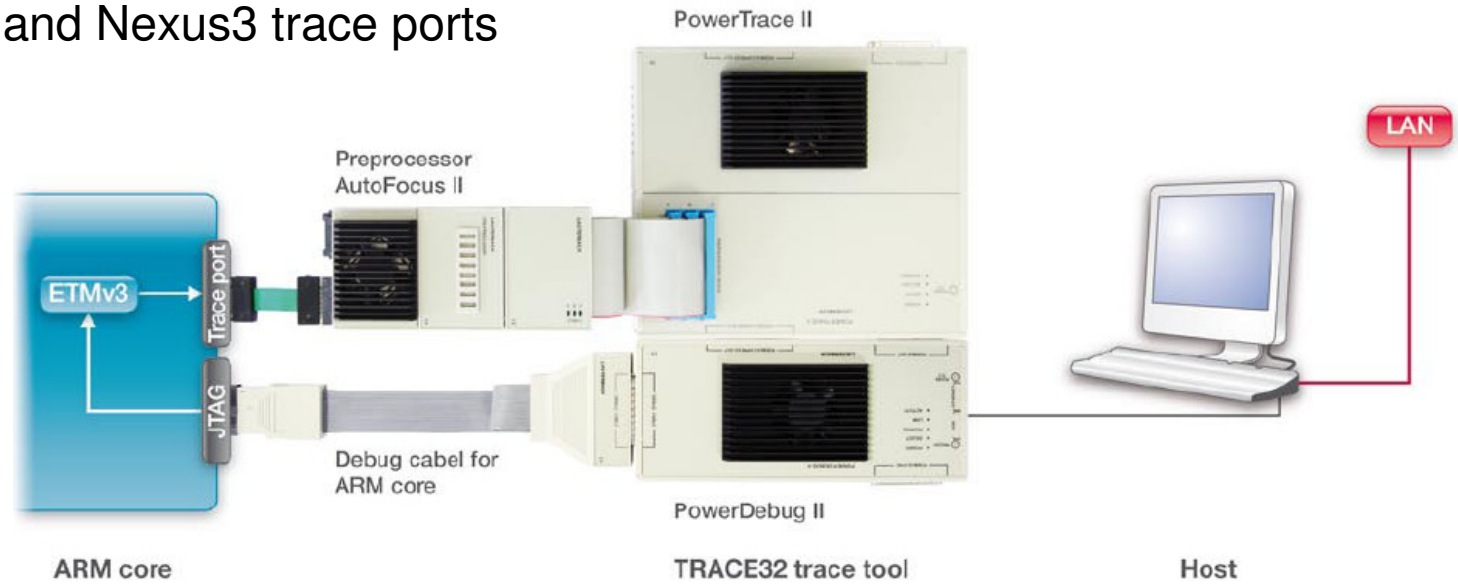
- Uses clock and signal pins
- ARM serial trace technology
- Tools can record up to 4 channels at 6.25 GBits/sec per channel

(could transfer entire contents of a DVD in 4 seconds)

Long term trace

(local demo available since January)

- Buffered and part processed on the fly by the trace tools
- Then streamed to a hard drive via one core of a multicore host
- Max trace speed 300mhz trace clock
- 500Gbyte drive gives 4 days of trace
- Available for ARM ETM and Nexus3 trace ports



SMP

- Tools available since early 2008 – in use with many customers worldwide

- Just need current release of our software and multicore licensed tools

- The cores most readily available are using silicon based on ARM , Mips, SH and PowerPC. Rtos types include WinCE, Linux, ThreadX, QNX.

- If you plan to use SMP – make sure the debug port has FULL access and control over ALL cores

- As I said before - - we know one processor which has supposedly been designed for SMP applications but which only has full debug control of the first core, core 0.

Summary

- The best debugger of all is between your ears
- If you use it before you start the project you can remove many potential problems