

Designing Complex Embedded Systems without an Emulator

With the increasing power and higher clock frequencies of the latest 32-bit RISC-based processor designs, a new debug and development strategy is required for embedded applications. Traditional approaches using an In-Circuit Emulator are becoming less practical. As a result, processor manufacturers such as Motorola have had to design debug support into their chip architectures. All their latest RISC designs - PowerPC, ColdFire and M.Core - offer various levels of built-in debug capability.

However, these on-chip resources need external hardware and software support to provide a complete development solution. This article demonstrates how the "Emulator Gap" can be bridged by combining the capabilities of Motorola's on-chip BDM interface with hardware tools support from Hewlett Packard (HP) and the software debug tools of Software Development Systems (SDS).

When faced with a seemingly intractable software problem, most embedded systems developers will reach for the emulator. Its combination of real-time trace, hardware breakpoints and low intrusiveness will generally make it obvious which part of the program is misbehaving when less hardware-intensive methods make it difficult to pinpoint the problem. However, this approach is rapidly becoming impractical with the new generation of highly integrated 32-bit RISC architectures.

One obvious problem is processor speed - devices are now routinely available at clock speeds of up to 200MHz. 300MHz will soon be common and will climb still further. This causes considerable practical difficulty for the design of the emulation connection. Emulators therefore become available much later - if at all - following silicon availability due to the difficulties these clock speeds present to the emulator designer.

This is not, however, the biggest problem. And in any case, not every embedded application requires the highest clock speed. Indeed, in many applications, a low speed may be desirable, due to power consumption constraints.

In fact, the main driving force behind the ever-widening Emulator Gap is the rapidly increasing level of on-chip integration - both I/O and memory. This is particularly noticeable in complex devices aimed at deeply embedded applications, such as automotive and communications systems. Such devices often have significant amounts of fast memory of various types on-chip, including cache, Flash, and RAM. This obviously offers several major benefits to the embedded designer - lower device count, less power consumption and reduced cost.

The problem is that, with so much on-chip, it becomes

very difficult, if not impossible, for an emulator to work out what is happening in the processor core. The logic analyser now has a problem too - since internal memory accesses may not be visible on external pins, there may be insufficient data available.

Indeed, in some designs aimed at the most deeply embedded applications, the pins carrying external address and data busses may even be eliminated - the remaining pins being dedicated to serving the on-chip I/O functions. Removing these pins does not just reduce package size - it also significantly reduces power consumption, since the on-chip bus drivers are no longer required.

So yet again the embedded systems developer can achieve a lower power, lower cost design - but at the expense of a further decrease in the visibility of on-chip processes to conventional development tools. Of course, this can have a major impact on project development time, since the hardware/software integration phase may be severely delayed until emulation tools become available. And in the case of the more com-

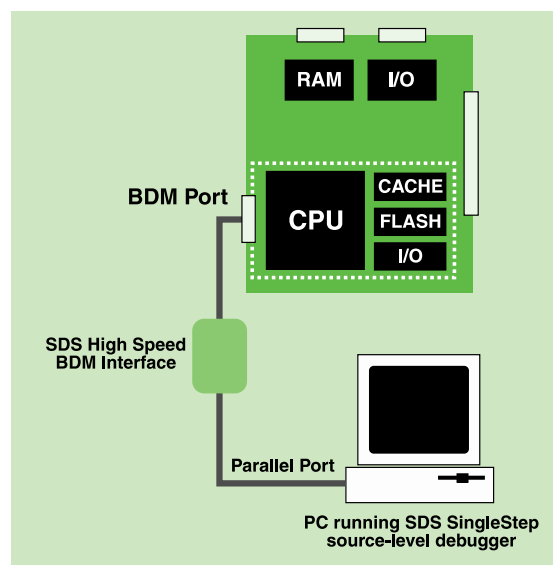


Figure 1. PC running SDS SingleStep source-level debugger

DEV. ENVIRONMENT

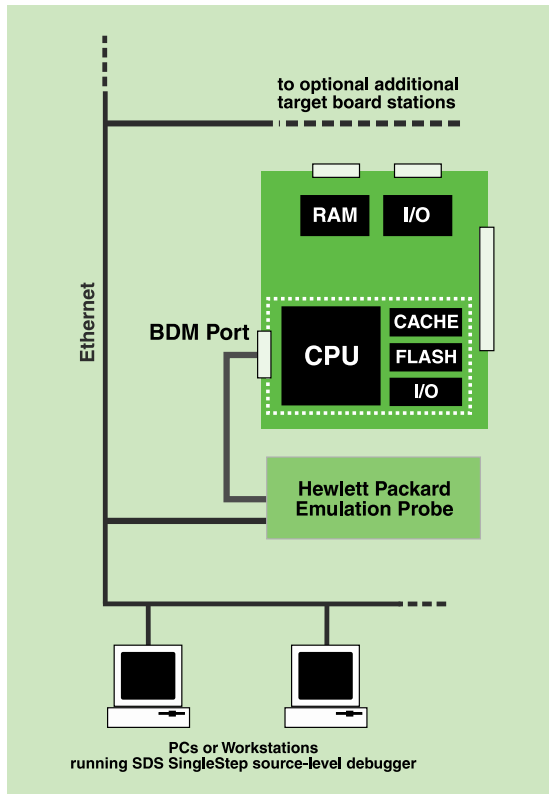


Figure 2. The on-chip debug control circuitry is closely coupled to the processor functions.

plex 32-bit architectures, there is no certainty that emulation support will ever appear i.e. the Emulator Gap may become infinite.

Motorola were one of the first processor manufacturers to identify this Emulator Gap problem and the first to provide a practical solution. Their range of 683xx microcontrollers incorporated Background Debug Mode (BDM) support on-chip. This was a simple interface giving the embedded developer a "static debug" capability for code download and run control, essentially providing the equivalent functionality to a target monitor, as well as an effective target connection mechanism.

This BDM approach has a significant performance advantage over the Emulator. The on-chip debug control circuitry is closely coupled to the processor functions and is using the same silicon; therefore its speed will increase with the speed of the processor clock. The Emulator, however, will have a fixed speed, and has the additional problem of bringing the data out to its own external control registers.

Motorola has greatly improved the capabilities of the BDM interface in later designs such as PowerPC, ColdFire and M.CORE. A key enhancement was hardware breakpoint support. In fact, for fast, highly integrated microcontroller designs such as the 860/PowerQUICC and the new MPC555, this may offer a better breakpointing solution than anything achievable with an emulator. Since the comparator for the BDM breakpoint is on-chip, its maximum "skid" is only a single cycle, whilst an emulator will have a performance several cycles worse, depending on the real-time bandwidth of the probe connection.

Better still, the on-chip BDM supports multiple comparators with Boolean operations that can provide complex conditional breakpoints and triggering, as well as non-intrusive counters which can be applied to these same comparators. Combined with the capability to monitor simultaneously the processor's internal data bus, many different types of real-time trigger conditions can be supported.

This does not just provide better real-time response than a conventional emulator design - which will always be dependent on mapping target memory to emulator memory - but it makes it possible to breakpoint code anywhere it may be located - even in cache memory.

Developers using this current generation of complex embedded devices normally require support for source code debug (typically C/C++) and it is obviously important that the debugger has been designed to integrate easily with these advanced facilities.

A further problem is that many embedded controllers now provide on-chip Flash memory, so the developer must get the code into the target device before debug can even start. Unfortunately, there are a multiplicity of protocols and algorithms for programming the different types of Flash memory. And the speed of the BDM interface itself may be a significant factor, if large amounts of on-chip memory need to be uploaded with revised code at each development iteration.

The source code debugger needs to provide support for these various methods, and - ideally - should offer a simple GUI interface which integrates the correct Flash algorithm into the code download process transparently to the user. Indeed, this capability could even be useful at final production programming and test, allowing both processes to be combined in one operation.

Although we have not yet covered all the functions of a conventional emulator, the level of debug capability described so far can be achieved at a tiny fraction of the cost of an emulator, using a simple BDM cable interface connected to the printer port of a standard desktop PC. And as long as this BDM connection device has sufficient bandwidth (>100Kbytes/sec or more) this simple configuration will provide a fast and responsive development environment for the majority of the debug problems encountered in embedded systems development.

However, some remaining development problems will still demand more advanced functions, such as trace and trigger facilities. The BDM approach can not only support these functions, but it also eliminates some of the problems encountered by the Emulation approach. The most serious is the need for bond-out silicon to be available. These are chips contained in large and expensive packages that brings out all the additional internal signals of the processor needed by the emulator.

However, since many of the more recent chips are no longer made available in bond-out form by the manufacturers, the Emulator may not have access to these internal signals. Indeed, with some modern chip packaging formats, such as BGA, there may not even be a

AD SDS

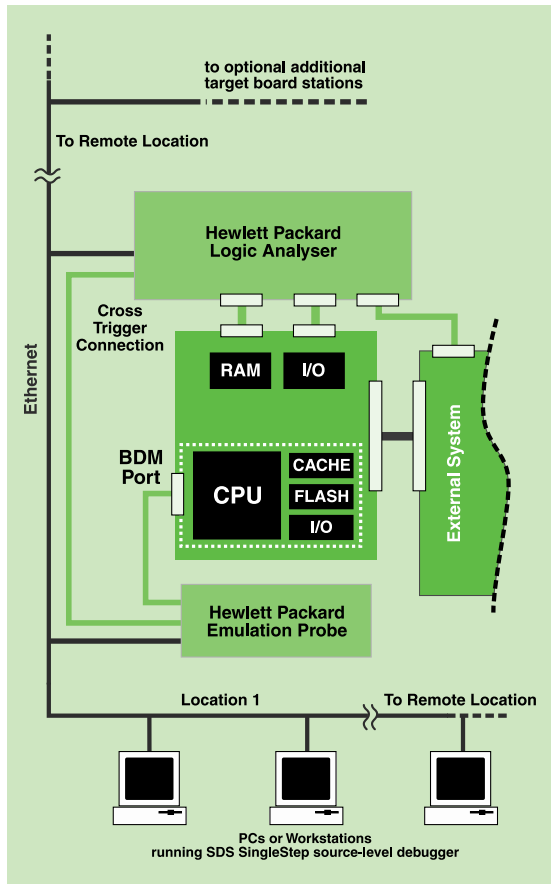


Figure 3. BDM supports multiple comparators with Boolean operators

simple way to connect the external signals to the emulator pod.

Even with bond-out parts available, similar visibility problems may also apply to specialised on-chip circuits such as cache and I/O functions that do not show their activity on the internal system bus. So instead of requiring bond-out silicon to support advanced debug capability, or needing elaborate (and expensive) pod connection schemes, a number of processor designs now allow access to these internal signals through the BDM interface.

To gain full advantage of these features, however, external hardware assistance is required. The first step is to use a more intelligent connection to the BDM interface, such as Hewlett Packard's Emulation Probe.

Since the on-chip BDM comparators can initiate a trace at any point in the code, it is possible to provide full trace/trigger capability, as long as the debug software is sufficiently intelligent to reconstruct the code execution order even through complex processor pipelining and cache operations, and in a non-intrusive manner.

The main limitation of this approach concerns the size and speed of the accumulated trace data. The BDM connection itself has limited bandwidth, so at high processor clock speeds it may not prove adequate to support the capture of the trace data.

This is a situation where a Logic Analyser is useful. The trace data can be captured at rates reaching GHz frequencies, into larger buffers than are typically sup-

ported by conventional emulators and on multiple channels if necessary. Furthermore, external boards and peripherals can also be instrumented, allowing collection of local trace data or providing trigger conditions of their own. However, to work effectively, this requires a real-time cross-triggering connection between the Logic Analyser and the BDM interface, since the Analyser itself has no run-time control capability. This is now provided by the Emulation Probe. In return, the Analyser can provide the Probe with the ability to trigger on off-chip events and signals.

This configuration, combining Source code Debugger, Emulation Probe and Logic Analyser will support even more extensive trace facilities. For example, complex watchpoint conditions can be used to output internal events on specific external pins, thus providing the ability to trace processor state changes. Better still, for advanced designs such as the PowerPC® 8xx/5xx families, the processor can be driven in a "show cycles" mode, externalising processor cycles that would otherwise be invisible to external hardware. As long as this capability is used intelligently by the debugger, by not forcing serialisation of pipelined instructions except at a change of flow of execution, for example, this can be used to facilitate full trace reconstruction with minimal intrusion on the real-time performance of the target system.

NEW PROCESSORS NEED A NEW DEBUGGING ENVIRONMENT

Implementation of this new approach to embedded debug and development requires the cooperation of key players in three segments of the industry. The silicon vendor, for example, must be willing to change their model to include on-chip debugging as an essential component of their chip designs. This does, of course, incur a penalty in terms of silicon space required - and more BDM features will require more silicon. But this is greatly outweighed by the advantages - making the once difficult task of connecting to the processor and establishing run control and register access a relatively simple and low cost effort. The hardware tools vendor can use the BDM port as a window into the operation of the processor via a target probe connection, providing a whole new range of capabilities for the systems developer and project teams.

The software tool vendor can then take advantage of this hardware assisted access to the processor by connecting the debugger interface to the port that allows processor run control and code downloading to occur under the control of the debugger interface. The software team now has a means to debug their code in-circuit using their familiar debugger interface. A logic analyser provides the real-time analysis component of the new architecture with adequate channels to view activity on multiple buses and external circuitry. Connection to the debugger interface from the software tool vendor provides access to source code symbols and allows the design team to access the logic analysis features through the common user interface of the debug software.

A NEW APPROACH - THE BENEFITS

Perhaps the greatest advantage of all in this new approach to embedded development is that exactly the same user interface and debug environment can be used across the entire design cycle by both hardware and software engineers. This will provide a major boost to the productivity of the entire development team, who can now share access to the target systems through the same GUI debugging interface, the same (networked) target connection, and the same hardware tools support.

This fresh approach opens up completely new ways of working for embedded system designers. Hardware and software engineers now have a concurrent development environment allowing a level of co-operation that was not possible when both groups worked independently. Now both groups can use exactly the same tools. This makes sure that they can work together at any state of the design and they talk the same language. The fact that developers have access to a common set of debug functions will make it easier to track down problems without changing tools halfway through a project simply to gain access to an extra facility, whether it is real-time trace or hardware breakpoints.

As all participants of the design team are now familiar with the complete tool chain, it is easy for the hardware engineer to collaborate with the software developer on driver design for a new peripheral. On a higher level, members of the design team can now work across the LAN in a "distributed emulation" environment, using the integrated Ethernet connection of the Emulation Probe. Each workstation now has a direct connection to the BDM interface of the target and the Analyser.

Using a source-level debugger such as SingleStep from SDS, which is fully aware of these real-time connections, the design team has seamless access to a distributed development environment which goes far beyond the capabilities of conventional Emulators. Thus, these powerful real-tracing and triggering capabilities can be harnessed directly to the software debugger to provide, for example, a source-code window allowing code execution to be viewed and correlated to these real-time measurements.

And by using this approach, both design groups are using the same tool, and are speaking the same language when discussing their problems. As there will be only one hardware setup with the Analyser, the test results will be the same. This eliminates the situation where the Emulator test results might be different when running the same tests using an Analyser.

And once the next project comes round, no need to wait for an Emulator - with just a simple software upgrade, the same Logic Analyser will be ready for the next project from day one. ■

Geoff Revill was appointed by SDS to the position of Vice President and General Manager Europe in May 1998. He joined the company in March 1995 with the task of setting up their European operations. As of January 1999, SDS have sales and technical support offices established in UK, France, Germany and Sweden, as well as an extensive network of distributors and VARs across Europe. Prior to joining SDS, Revill was UK Sales Manager at ISI responsible for RTOS products. Geoff holds an honours degree in Computer Science.



Figure 4. SingleStep PowerPC 8xx/5xx - BDM Interface