Simulation of SOC (system-on-chip) design complexity with a full range of real-world stimuli proves impossible in today’s tight design schedule. The system requires advances in verification methodology. Additionally, new designs have a wider diversity of elements than ever before and often draw upon modules from multiple sources. The nature of designs demands that you perform internal functional and timing verification with a high degree of certainty to determine performance accuracy. Finally, the devices fit on a board with other components, which requires a verification method that combines both internal and board-level debugging methods.

New debug strategies combine on-chip and off-chip tools

Microprocessors add on-chip debug

The complexity of microprocessors and microcontrollers presents a unique set of verification challenges. To add further complications, critical signals are embedded inside the chip with no outlet to an I/O pin. In response to these challenges came the creation of on-chip debugging tools that control operations and access information inside the devices to isolate problems during design development. These tools typically involve interaction with on-chip circuitry via a dedicated set of I/O pins, including Motorola’s BDM (Background Debug Mode) and OnCE (On-Chip Emulation), MIPS’ Enhanced JTAG, and NEC’s N-Wire.

These processor-specific debugging tools typically allow activities, such as resetting, stopping, single-stepping, setting breakpoints, and examining or modifying register contents, memory contents, or both. The tools also allow capture of addresses at system speeds for program-counter traces to a maximum clock frequency (generally 100 to 200 MHz).
With an increase in processor integration onto system-level chips, it stands to reason that the debug circuitry should follow. However, SOCs have requirements beyond that of single processors, including the development of new debugging technologies. First, SOCs contain a wide range of elements beyond processors that can benefit from specific debugging features, including DSP and other mathematical operators, communications functions, and specialized I/O or bus interfaces. Also, the increase in use of IP (intellectual property) from diverse sources demands an especially rigorous verification on the level of the chip’s individual functions. Because speed is at such a premium in many of today’s systems, capturing data at actual system speeds is a must. Finally, the need for high-pin-count devices, such as ball-grid arrays, combined with the decreasing amount of board space, favors a solution that avoids external probe points.

**EMBEDDED-LOGIC ANALYSIS**

Ideally, embedded-logic analysis should minimally affect a design’s I/O and internal resources and its design flow. As with embedded processors, embedded-logic analysis for SOCs often relies on streaming captured data out of the JTAG pins that are in place for boundary scan, device configuration, or both—minimizing the need for extra I/O. Of course, you can allocate extra I/O as needed for captures of wide data samples. The logic required for embedded test is negligible—in many cases less than 1% of the overall design.

Greater device capacity and features, increasing IP availability, and tool advancements have led to SOPCs (systems on a programmable chip). With SOPCs, embedded analysis carries additional implications because of the hardware flexibility.

In programmable logic, the embedded-logic-analysis circuitry uses some of the same logic and memory resources that you would use to implement your design. The amount of resources you use is a function of the number of internal nodes you probe and the number of samples per node. Altera’s SignalTap embedded-logic-analysis technology provides some sample numbers for the exact amount of resources needed in the Apex device architecture (Table 1).

In the Apex product family, LEs (logic elements) provide the logic-implementation capability, and ESBs (embedded-system blocks) provide embedded-memory resources. As an example, the Altera EP20K400E has 16,640 LEs and 104 ESBs available. Based on these numbers, a sample embedded logic analyzer in an EP20K400E design requiring 32 sample nodes with a sample size of 1024 samples per node requires less than 3% of LE resources and 15% of ESB resources.

In custom devices, you can determine the size of the memory buffers required for capturing sample data. In programmable logic devices, you can set the embedded-memory structures that house the captured data to the depth you need for test requirements. As a result, the total amount of embedded-memory resources limits the amount of captured data. The granularity of the embedded-memory blocks impacts the amount of memory available to your design. For example, a single large-scale PLD might have N blocks of embedded memory, one of which you might use for capturing data. Regardless of the amount of memory block you use for data capture, only N-1 memory blocks are left for your design.

In the design flow, embedded-logic analysis should be as unobtrusive as any other design element. Designers need only to decide which design points should be probed and define their triggering conditions; the associated tools should focus on facilitating that process and removing any other burdens associated with the debug activity. Ideally, building the embedded-logic-analysis function should be part of the design-development tools themselves, because ver-

### Table 1—The Logic and Memory Use of Embedded-Logic Analysis in Apex Devices

<table>
<thead>
<tr>
<th>No. of nodes to be sampled</th>
<th>Logic elements needed</th>
<th>Embedded-system blocks needed</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>128 samples per node</td>
</tr>
<tr>
<td></td>
<td></td>
<td>256 samples per node</td>
</tr>
<tr>
<td></td>
<td></td>
<td>512 samples per node</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1024 samples per node</td>
</tr>
<tr>
<td></td>
<td></td>
<td>2048 samples per node</td>
</tr>
<tr>
<td>One</td>
<td>136</td>
<td>One</td>
</tr>
<tr>
<td>Two</td>
<td>144</td>
<td>One</td>
</tr>
<tr>
<td>Four</td>
<td>160</td>
<td>One</td>
</tr>
<tr>
<td>Eight</td>
<td>192</td>
<td>One</td>
</tr>
<tr>
<td>16</td>
<td>256</td>
<td>One</td>
</tr>
<tr>
<td>32</td>
<td>384</td>
<td>One</td>
</tr>
<tr>
<td>64</td>
<td>640</td>
<td>One</td>
</tr>
<tr>
<td>128</td>
<td>1152</td>
<td>One</td>
</tr>
</tbody>
</table>

**Figure 2**

START OF FRAME

```
1. PARSE
   00010111
   STATUS
2. TRANSLATE
   01001100
   CONTROL
   73FB
   A-TO-D
   D-TO-A
   INTERNAL
   EMPTY
   OVERFLOW
   IDLE MODE
   INTERRUPT
   CLOCK
   EN MODE 1

   23 dB
   25 dB
   OK
```

END OF FRAME

Software associated with the external analyzer parses and decodes the CAN packet.
ification is a significant part of the design flow. After you build the analysis function, you must connect the analysis elements to the design elements so they are testing in the design description; there is no reason that an intelligent development tool could not perform this connection transparent to the user. Furthermore, any changes you make to the conditions of embedded analysis that do not require logic or connectivity modifications (that is, changes in triggering conditions and modifications of data-sample size) should not force a new layout or a new place-and-route of the design.

EMBEDDED-LOGIC AND BOARD-LEVEL ANALYSES

The combination of embedded-analysis circuitry and onboard memory allows data capture at system speeds for SOCs and SOPCs; however, these components do not exist in a vacuum. The boards they occupy also require debugging, and in many cases, the board-level debugging must be time-correlated with the internal debugging for a thorough evaluation of the entire system. Such correlation requires interaction between the embedded test circuitry and the external analysis equipment. This interaction should allow flexible cross triggering in which you can use any combination of internal and external data to trigger the capture of either internal or external data, including multilevel triggering. Furthermore, the cycle of defining triggers and capturing data should be an interactive, real-time process to maximize the efficiency of the verification cycle. Particularly useful for the proliferation of high-speed communications systems is the ability to capture data at high sample rates into deep memory buffers.

PARTICULARLY USEFUL FOR THE PROLIFERATION OF HIGH-SPEED COMMUNICATIONS SYSTEMS IS THE ABILITY TO CAPTURE DATA AT HIGH SAMPLE RATES INTO DEEP MEMORY BUFFERS.

Quartz, creates and embeds a logic-analysis module into a PLD design based on user input and data-capturing conditions. The data is captured in real-time inside on-chip memory blocks and streamed out of the device via the JTAG port (Figure 1). The data is then sent to a host computer running the Quartus software via the external analyzer. The analyzer contains probes for capturing board-level data, and each probe has a 1 Mbyte-memory buffer for capturing deep samples. The externally captured data can trigger data capture inside the device and vice versa; single events, patterns, pattern duration, time-outs after events, and multilevel triggering with as many as four events can trigger data capture.

To support hardware-platform independence, you can move SignalTap Plus embedded-logic analyzers from their programmable-logic options to a custom component option. Doing so requires the same test setup (that is, the external analyzer and the Quartus software), and the embedded-logic analyzers themselves simply become a part of the overall design layout.

FURTHER APPLICATIONS

Capturing internal SOC and SOPC signals in real time is an advantage of the debugging process; however, advancements in this technology reduce the verification burden. Captured data itself is only a series of waveforms, usually large in number, based on the million- and multimillion-gate devices in today’s designs. A tool that decodes those waveforms into higher levels of abstraction would be an additional advantage to engineers. For example, Hewlett Packard’s N-Trace technology accomplishes this function for embedded processors (Figure 2). N-Trace uses embedded-test circuitry to capture real-time processor data, such as the number of instructions executed and branch-source and desti-
CAPTURING INTERNAL SOC AND SOPC SIGNALS IN REAL TIME IS AN ADVANTAGE OF THE DEBUGGING PROCESS BUT REDUCES THE VERIFICATION BURDEN.

This data is either captured in a memory or a pipeline before it is streamed out of the device via a small, dedicated debug port, where an external logic analyzer picks it up. Associated with the logic analyzer is a software program that can decode the data into a form that allows easy recognition of the processor’s internal operations.

Designers should be able to apply decoding technology more broadly to SOC and SOPC designs. As a result, designers would see not only higher-level abstractions of their processor’s internal operations but also other functional blocks on the chip. For example, Figure 2 shows the parsing and decoding of a CAN (Control Area Network) into its individual elements (start of frame, data, and cyclic redundancy check).

To unleash the full potential of this decoding/abstraction process, the mechanisms of the decoding process should be exposed in a parameterizable or scriptable format, allowing users to define their own levels of translation and abstraction. In this way, any given set of captured data streams could be displayed in any format, including hex, ASCII, analog, and decimal. This function would be useful in decoding and debugging communications streams, such as Ethernet packets or ATM cells; interface transactions, such as those in a USB or PCI bus; and embedded-processor instructions.

Testing and debugging have always been important parts of the design flow, but in an era of increasing device sizes and complexity, increased use of IP from multiple sources, and intense time-to-market pressures, verification looms as the most challenging part of the design process. The trend towards greater integration of entire systems into single devices extends to the testing and debugging requirements of those systems. Embedding debug tools onto the chip will allow test methods to maintain technological parity with the system under test, and by adopting them, chip designers will assist the system-integration process. Furthermore, embedded analysis in coordination with external test tools are the key elements to debug and verify system-level chips and the boards they occupy. Without this class of tools, debugging complex SOC and SOPC designs will be nearly impossible. Advances in test and verification will allow systems designers to reap the full benefits of SOC and SOPC methodology.

**References**


**Authors’ Biographies**

Bob Garrett is marketing manager for Altera Corp in Santa Cruz, CA. He was formerly director of marketing for Boulder Creek Engineering, acquired by Altera in May of 1999 and has over 20 years of experience in digital test and measurement systems.

Martin S Won is a senior technical staff member at Altera Corp. He has over nine years of experience in digital system designs involving programmable logic. He holds a BS in Electrical and Computer Engineering from the University of California at Santa Barbara.