For reasons of cost, performance, power, and miniaturization, many electronic systems that once consisted of several printed circuit boards are now manufactured as a single semiconductor device. As a result of this complexity, previously accessible system signals can no longer be observed, and the process of isolating and analyzing silicon problems has become tedious and time consuming.

Using a process called silicon debug, engineers try to locate the sources of errors in their devices. They work to understand the root cause, correct or work around the root cause, and prepare the device so high-volume manufacturing can proceed.

Engineers currently use various software and hardware tools to debug silicon. But because failure modes are complex and often arise only in corner-case scenarios under at-speed operation of the system, external debug equipment is insufficient to track down the source of system failures.

Design-for-debug (DFD) methodologies offer a better solution. By embedding DFD logic right on a chip and then stimulating and evaluating it with commercial silicon debug and analysis tools, device manufacturers can make it possible for test engineers to view signals on a chip and track down the causes of failure.

Using DFD logic

By adding DFD logic to chips, designers make it possible for test and product engineers to observe key internal signals and use a host-side commercial debugging tool to analyze and understand nonconformant chip behavior. One of the most popular silicon debug techniques is state dumping. This technique can be implemented using a standard on-chip DFD architecture that consists of only three debug features, which we call the “ABC” of silicon debug. An IEEE 1149.1 standard Test Access Port (TAP) controls all three features:
--Access to the internal scan chains. Reusing the internal scan chains, which are typically already inserted to facilitate chip manufacturing test, is a common way to enable easy debugging. Reusing the scan chains is inexpensive and provides the complete state of the chip when scanned out via the TAP.

--Breakpoints. When an important, internal event or sequence of events is detected, an on-chip breakpoint module stops the functional clock signals. This breakpoint is programmable via the TAP.

--Clock control. When the functional clock signals have been stopped, an on-chip switch is programmed to have the functional clock signals follow the TAP clock during scan out.

The resulting debug architecture scales well with the chip’s number of clock domains and flip-flops, and we have found that its cost in terms of silicon area is typically well below 1% of the total chip area. The DFD logic provides full access to the internal signals at predefined points in time, while the chip is operating in its intended application board.

Figure 1. This application setup uses a debug architecture to obtain state data from a chip embedded in a system.

Figure 1 shows the setup we use to debug silicon. The setup uses the debug architecture to obtain state data from a chip embedded in a system. A desktop computer is connected to the on-chip TAP controller. Normal user-defined TAP instructions control this setup.

When using such a setup for your own debug application, your first step is to program the breakpoint, or the point in time when you want to take a snapshot of the circuit’s state. Then, the chip is functionally reset and initialized in its environment to a known, stable state. The application starts to execute, and when the breakpoint is hit, the breakpoint is activated and the on-chip functional clocks stop.

The debugger software running on the desktop computer detects the stopped clock, switches the circuit into debug scan mode, and extracts the content of the on-chip scan chains. The resulting bit stream (referred to as a state dump) is stored on disk for further processing and interpretation. In most cases, this post-scan dump-data operation is performed by the debug system itself.

Your next step is to prepare the data for use with host-side debug systems. The software maps the data (which consists only of ones and zeroes) to its origin in the design and system time as follows:
1. For each scan data extraction, the software adds a time stamp or cycle number based on the breakpoint that was hit;

2. The software associates the data with the gate-level signal names in the HDL design; and

3. The software subsequently writes the data in a format such as the Value Change Dump (VCD) standard. As an alternative to gate-level mapping, you can associate the data with signals at the register transfer level (RTL). Hardware designers trying to understand the behavior of their silicon often prefer this level of design abstraction.

**Expanding the data**

The scan dump is formatted into a VCD file. To allow you to observe internal signals not accessible via the DFD logic (for example, values are in the combinational logic clouds between the scan registers), the software computes the missing data through a process called “data expansion.” In this process, the software

1. reads the mapped fan-in-cone scan register values corresponding to the internal signals to be observed;

2. sorts the combinational logic between these registers using a linear ordering algorithm;

3. computes the values for these signals using a cycle-based evaluation; and

4. creates a VCD file with the computed values.

The result of data expansion is that all computable combinational signals contain a Boolean value of 1 or 0 (click to view Figure 2). If some of the state values are not available (that is, are not provided in the scan dump), some circuit values cannot be computed. In the generated views, these values are denoted with the special value “NC” (not computed). The computed values from data expansion, along with the scan dump values, are passed to a debug system via the VCD file.

Although data expansion can be considered similar to simulation, it is not burdened with the computational overhead that a simulator requires for time processing. You could further optimize data expansion by dynamically generating the combinational signal values for display in a debug system rather than by using a VCD file. Usually, software-based debug systems read VCD files dumped from pre-silicon design simulations to trace the root cause of errors detected prior to tape-out. Using VCD files containing information obtained from actual running silicon, and created using the steps described above, effectively makes the debug of silicon the same as for simulation.

Because most designers are familiar with their simulation-centric verification environment, they can now immediately assist with silicon debug. Designers can apply pre-silicon techniques to isolate the root cause of any problem, all while operating in their familiar tool environment. The standard capabilities available in debug tools include the ability to perform source tracing, waveform viewing, schematic viewing, and value annotation on any of the design views. If available, expected data based on golden simulation results can help narrow down an error’s root cause by highlighting differences with actual silicon data.

**Case studies**

For a set of typical use cases, we measured the effectiveness of the data-expansion method. We selected modules of two real designs and dumped out all register values and input values of the designs. Then, we applied the data-expansion capability available in the Novas Siloti SilVE visibility enhancement tool to calculate the missing values. Table 1 shows the results.
While the size of the circuits varied significantly, the percentage of observable signals did not. The observable signals ranged from the mid-90s to 100% of all the signals. With this methodology, you can observe nearly all of the signals by dumping out only a limited number of registers and primary inputs for data expansion. This methodology also keeps the area impact to a minimum by leveraging already existing design-for-test (DFT) resources that would otherwise be idle during system-level debug.

Overall our approach bridges the traditionally separate environments of chip designers and silicon-validation engineers. The DFD logic can be readily used on all digital chips. The application allows engineers to share knowledge from both domains to find and resolve design errors in silicon more quickly and easily, yielding shorter time-to-market and higher product quality.

Table 1. Data expansion results

<table>
<thead>
<tr>
<th>Case</th>
<th>Total Signals</th>
<th>Dumped Signals</th>
<th>Dumped-To-Total Ratio</th>
<th>Expanded Signals</th>
<th>Observed Signals</th>
<th>Observed Signal Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Circuit 1</td>
<td>347</td>
<td>110</td>
<td>32%</td>
<td>237</td>
<td>347</td>
<td>100%</td>
</tr>
<tr>
<td>Circuit 2</td>
<td>32,009</td>
<td>4,593</td>
<td>14%</td>
<td>25,560</td>
<td>30,153</td>
<td>94%</td>
</tr>
<tr>
<td>Circuit 3</td>
<td>138,842</td>
<td>13,190</td>
<td>9.5%</td>
<td>125,652</td>
<td>138,842</td>
<td>100%</td>
</tr>
<tr>
<td>Circuit 4</td>
<td>4,568,074</td>
<td>173,544</td>
<td>3.4%</td>
<td>4,356,475</td>
<td>4,530,019</td>
<td>96%</td>
</tr>
</tbody>
</table>