

Using Complex Triggers in the Identify Debugger

You can obtain huge productivity gains with Synplicity's powerful and comprehensive FPGA debug tool.

by Dennis McCarty
Technical Marketing Manager
Synplicity, Inc.
dmccarty@synplicity.com

Hardware debuggers represent the ultimate system verification tool. Unlike simulators, debuggers show what the logic is actually doing inside the device while running in the system at full speed. When using a hardware debugger, it is crucial that you capture the precise data you need to discover bugs and verify system behavior. Not only must you locate the logic transitions around a certain event, you must also track bugs that may be rare events and trap them for closer examination.

The Identify RTL debugger from Synplicity offers you a view of logic behavior inside an FPGA operating within the system. It also offers a highly sophisticated set of trigger mechanisms and other features that you can use to isolate events germane to a particular problem.

In this article, I'll describe some of the features of Identify.

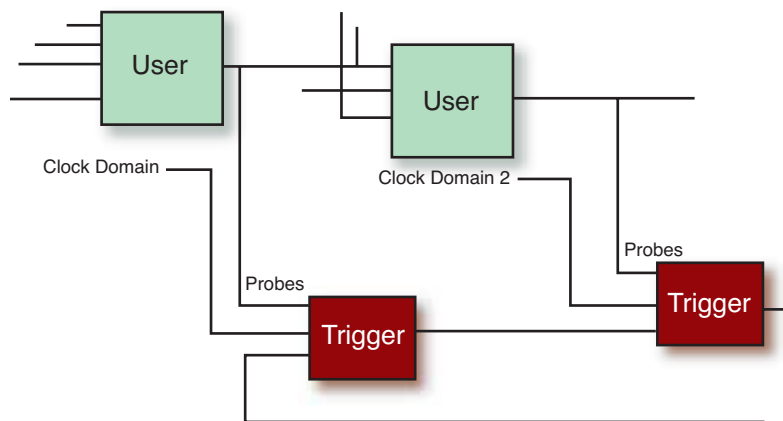


Figure 1 – Cross-trigger example

Triggering Across Clock Domains

Today’s FPGA designers frequently use multiple clocks, as these devices come with numerous dedicated clock buffers. In multi-clock systems it is common to encounter timing problems related to clocking data between domains. Such problems include metastability, failure to meet setup or hold times, and dropped data. Detecting these often subtle problems is usually difficult. The problem may not appear in logic simulation at all, and may only be detected while debugging by over-sampling within a domain or by triggering from one domain and sampling in another.

Cross-triggering is a technique that enables you to trigger on an event in one domain and sample an event in another. As shown in Figure 1, the Identify product allows the trigger logic of one domain to drive and enable the trigger in another. You can use cross-triggering to view the timing of events that cross domains. You can also use it to see events occurring within a clock period by over-sampling the period with a faster clock.

Sampling Modes

Sampling modes control the way data is added to the buffer when a trigger condition is reached. These modes allow you to sort data inflows by mode and increase buffer efficiency by storing only relevant data.

Identify software offers four sampling modes:

- The normal mode fills the buffer completely in a single trigger event. Subsequent triggers are ignored unless you run the debugger again.
- In the always armed sampling mode, the buffer fills on every trigger until the debug is stopped using the stop icon.
- The qualified fill mode stores a single sample on each trigger. The buffer will contain only events that caused a trigger and will continue until the buffer is full or when sampling stops.
- The qualified interrupt sampling is like qualified fill, except that sampling will continue until it is interrupted. If sampling continues after the buffer is full, old data will be overwritten.

The qualified and always armed sampling modes must be enabled separately for each intelligent in-circuit emulator (IICE) module during instrumentation. You can enable these modes by clicking on the IICE configuration button in the Instrumentor and checking the boxes in the IICE sampler menu, as shown in Figure 2.



Figure 2 – Sampling modes

The sample mode is set during debugging using the pull-down sample mode icon menu, as shown in Figure 3.



Figure 3 – Sample mode pull-down menu

Trigger Modes

Trigger modes control the way data is added to the buffer upon reaching a trigger condition. There are four operating modes:

- The cycles mode triggers on the number entered in the value field representing the number of clock cycles after the condition.
- The events mode triggers on the n th instance of a trigger condition. In this mode the value field specifies the instance.
- The pulsewidth mode triggers after the trigger condition has remained active for n clock cycles.
- The watchdog mode triggers when the condition has not been active for n clock cycles since the last trigger event.

The default mode is cycles. To use the other modes, you must enable them by selecting the IICE configure button and clicking on the “complex counter triggering” box under the IICE controller menu. Use the arrow selectors to set the counter width to the maximum binary value you might need (Figure 4).



Figure 4 – Enabling trigger mode

To select trigger modes, use the down arrow, as shown in Figure 5.



Figure 5 – Specifying trigger mode (pulsewidth mode selected)

Bus Trigger Expressions

The Watchpoint setup display is used for single-bit data (see Figure 6).



Figure 6 – Watchpoint setup

Setting the trigger for a bus or a portion of a bus is more complicated, but offers a more powerful form of triggering. A right-click on a bus brings forth the menu shown in Figure 7. Several values or ranges of values are available. Entering a value in the left column but not the right causes a trigger on the exact value. Entering data in both columns will cause a trigger on the transition from the left value to the right value. To enable the trigger, check the box(es) next to each one.



Figure 7 – The four values 0-3 indicate that the currently selected IICE was configured for state machine triggering and that the four values correspond to C0-C3 in the state editor.

Partial Bus Trigger Values

Partial bus instrumentation is the definition of one or more bits of a bus such that it can be instrumented separately. Partial bus segments are defined using the menu, which you can invoke by right-clicking on the bus and selecting “add partial instrumentation.”

Each partial bus segment can be instrumented using the bus trigger menu displayed in Figure 8.



Figure 8 – Instrumenting partial bus segments

Trigger State Machine Editor

The most precise and powerful way to detect a unique condition is to use a state machine as a trigger. A state machine can traverse between states on any condition and trigger, or not, in any state. By using a state machine, you can create a sequence of steps and conditions that must be completed to arrive at a trigger condition. The Identify tool includes a state machine editor that allows you to graphically tailor the steps necessary to create the exact trigger condition you desire.

Although it is certainly possible to create a state machine directly in the source code for the purpose of triggering on an event, the Identify editor automates this process by providing a menu-based method. Moreover, a manual solution would require that you manually adjust the logic and specify new trigger nodes during instrumentation for each trigger adjustment and re-synthesis.

Adjustments such as whether to trigger on a state, under what conditions, and how the counter will be used to trigger are made in the debugger. You can dynamically make these adjustments during debugging without tampering directly with the design, making it easier and more efficient to use the Identify product’s integrated graphical state machine solution.

Configuring the IICE for State Machine Triggering

Configuring the IICE in advance is required for state machine debugging. The state machine trigger submenu is located in the IICE configuration menu, as shown in Figure 9. After specifying state machine triggering, you use the



Figure 9 – State machine triggering through IICE menus

wheel switches to dial the number of states, number of trigger conditions, and the width of the counter. You do not have to use all of the resources specified at this stage during debugging.

Saving the IICE selection allows you to specify the behavior and triggering conditions when you are ready to debug. It is in the debugger where you define the state machine states and conditions. For any IICE that has been set to allow state machine triggering, an icon appears, as shown in Figure 10.



Figure 10 – Example of IICE module not enabled for state machine triggering

Those IICE modules not enabled for state machine triggering are shown with a gray box icon.

Defining the State Machine

Selecting the state machine icon invokes the state editor, as shown in Figure 11. The editor initializes to display a space for each of the states specified in the IICE configuration.



Figure 11 – Invoking the state machine editor

The editor has a pull-down insert macro selector from which you can select one of eight macros. The macros apply either one of the four trigger modes described above, one of two conditional modes, or one of two sample modes similar to those in the state machine.

Selecting a macro from the menu invokes the macro editor, which is used to define the macro function. The macro editor contains fields that determine which condition will be used for the state and the number of events or samples that will be counted. Select the condition(s) from among the numbered C values.

The Identify product brings uniquely powerful and comprehensive capabilities to FPGA debugging. The multiple clock triggering feature allows you to see events that are likely to remain undetected in a simulation environment.

Watchdog Timer Mode

The `st_watchdog` editor is shown in Figure 12 as an example. The editor defines the macro function and definition fields. Enter the transition condition in the A field. The transition is one of the state names among the number of states defined during instrumentation. The value for N is the number of clocks the timer counts before the trigger.

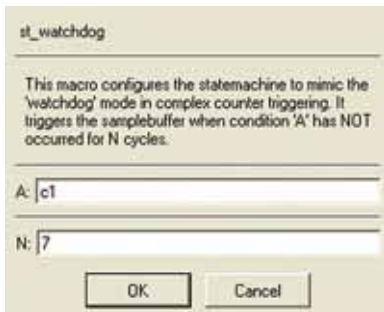


Figure 12 – `st_watchdog` editor

Conditional Modes

Two other macro examples are shown in Figure 13. On the left is the `st_B_after_A` macro. Here you enter two conditions (A and B) with the trigger based on the *n* number of times that B occurs after A has occurred. Condition A is then the qualifier to check for B one or more times for the trigger.

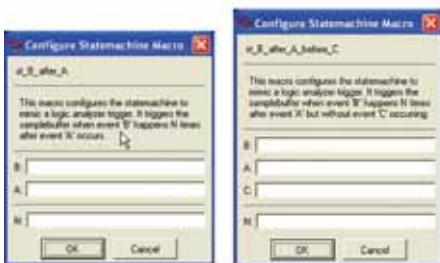


Figure 13 – Using conditional modes

State Editor

Each state has conditions under which it will transition to another state. The transition editor is used to describe the conditions of

one or more transitions from a state. You can invoke the editor by clicking on the pencil-and-paper icon. The editor includes fields and options for each state (Figure 14).



Figure 14 – The transition editor describes conditions of transitions from a state.

State Transitions

The first selection is the state number, from which the current state will transition. Use the thumbwheel to select the state. When you click OK to leave the editor, leave the “from” set to this state. If you select a “from” state other than the state where the editor was invoked, it will apply your changes to the other state and eliminate the transition altogether from the state you are editing. Remember, you can have any number of transitions to other states or remain in the current state.

Describing Conditions

In “on condition,” you specify the state condition under which the trigger will fire. The choices include any of the conditions (notated by a C) defined during the IICE configuration. These conditions are defined during instrumentation. Editing the value for any Watchpoint will display a value for each condition. Defining multiple Watchpoints as conditions will logically AND the conditions.

The default condition is “true,” meaning that the trigger will fire simply by entering the state. You can enter any of the C numbered values or “cntnull” by

typing in the value and negate the preceding value with an exclamation point.

State Machine Actions

The “actions” section works with the previous selections to allow another level of trigger control. The red T trigger box enables the trigger to fire when checked and when the previously described conditions exist. The remaining boxes control the counter and only affect triggering when the condition is selected as “cntnull.” That is when the counter reaches a value of zero.

The counter always decrements as represented by a counterclockwise arrow. The counter can be loaded to any value, as indicated by the down arrow. In any state the counter may be loaded, or enabled, to count down. If the counter reaches zero, it must be reloaded before its next use.

Checking the initialize counter box and entering a value starts the counter from that initial value. The trigger will, if enabled, fire when the counter rolls over.

You can add any number of additional state transition conditions to each state.

Transition values are cleared using the blank sheet icon. Transitions themselves are deleted using the X icon.

Conclusion

The Identify product brings uniquely powerful and comprehensive capabilities to FPGA debugging. The multiple clock triggering feature allows you to see events that are likely to remain undetected in a simulation environment. The sampling modes maximize buffer efficiency. The advanced triggering capabilities are a means for highly sophisticated refinement of data search methods.

The Identify product is a dynamic, in-system debugging environment that offers huge productivity gains, allowing you to debug in RTL code.

For more information, visit www.synplicity.com/products/identify/index.html.