Agilent 16760A Logic Analysis Module Triggering

Product Note

Acquisition Modes

The Agilent 16760A Logic Analysis Module supports the following six acquisition modes:

- 200 Mb/s State Mode
- 400 Mb/s State Mode
- 800 Mb/s State Mode
- 1250 Mb/s Half Channel State Mode
- 800 MHz Conventional Timing Mode
- 400 MHz Transitional / Store Qualified Timing Mode

The primary reason for differentiating the four state modes is that triggering capabilities decrease at each of the specified data rate boundaries. Users whose system data rates allow operation in one of the lower speed modes gain greater triggering capability at each state speed boundary. At 1250 Mb/s the analyzer channel count is also reduced by half.

The 200 and 400 Mb/s state modes and the two timing modes provide the same triggering capabilities available in all other VisiTrigger family analyzers at equivalent speeds with one exception - the 16760A does not allow setting actions for the flag bits in either timing mode. Some of the limitations and capabilities of the 200 Mb/s and 400 Mb/s modes are discussed starting on page 11.

VisiTrigger and the Trigger Compiler

Triggering in the 16760A logic analyzer is setup using Agilent's VisiTrigger interface. When a measurement is taken (by pressing the run button), the specified measurement conditions are translated by the trigger compiler to actual hardware pattern resources, combiner resources, etc. The VisiTrigger interface does not enforce a strict set of limits that guarantee that the specified measurement can be compiled into hardware because:

1. The user may temporarily make a change that would violate one of the rules, but correct it later. For example, adding a pattern resource that would put the number of pattern resources over the limit, but then removing a pattern resource in another location. Strict enforcement would require the user to first delete the undesired resource before adding a new one.
2. Since the trigger interface does not closely resemble the hardware actually implementing the measurements, it is difficult to easily predict when some resource limits have been exceeded. The interface would essentially have to execute a compile after each user change to determine if any limits have been exceeded. This would cause the responsiveness of the interface to become unacceptably slow. Interdependencies that exist between different types of resources would also make it impossible for the interface to determine a specific change that will correct the problem.

This means that the interface will allow the user to enter measurements that will not compile. When the measurement does not compile, the user must change the measurement setup in ways that will allow it to compile.

The purpose of this document is to present the rules and guidelines that represent the actual limits of the hardware and compiler with examples illustrating compilable measurement constructions.

Labels

The 16760A module provides 17 channels per pod (16 plus one 'data on clocks' channel). Each analyzer card provides two pods (34 bits per card).

A label can contain a maximum of 32 channels. A label that contains channels assigned from more than one pod is referred to as a split label. Labels that contain only assigned channels from a single pod are non-split labels. In the example in figure 1, labels Pod_C1, Pod_C2, Pod_D1, Pod_D2, and Pod_D2wClk are non-split labels; the remainder are split labels.

The type of label (split or non-split) affects some of the triggering capabilities in all modes. The effect becomes more apparent as state speed increases and the effective number of

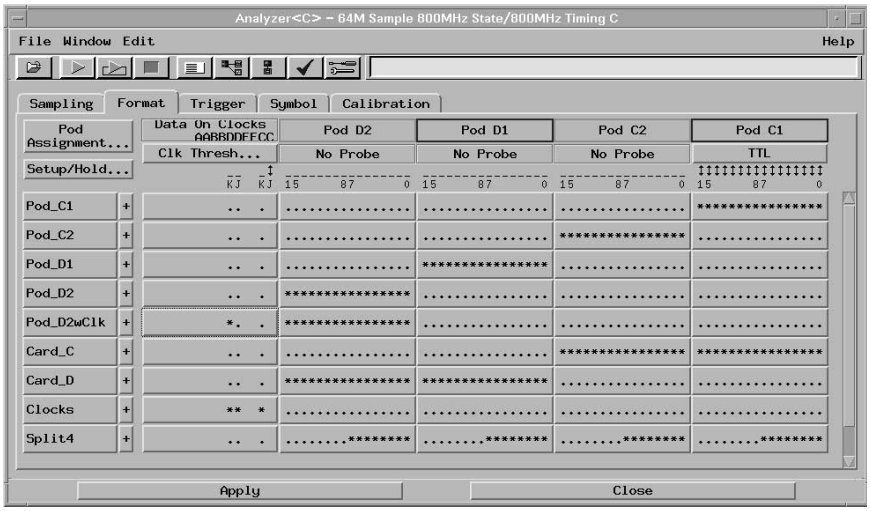


Figure 1. Logic analyzer inputs are assigned to labels in the format menu.

800 Mb/s State Mode

In 800 Mb/s state mode the following 8 trigger functions are available:

- Find pattern
- Find 2 patterns in immediate sequence
- Find 3 patterns in immediate sequence
- Find 4 patterns in immediate sequence
- Find 2 patterns in eventual sequence
- Find 3 patterns in eventual sequence
- Find 4 patterns in eventual sequence
- Run until user stop

Only one of these functions may be selected at any one time; sequencing of the functions is not allowed (see Figure 2).

Each of the functions allow modifications such as ANDing or ORing multiple events, changing the operations on the pattern (=, ≠, <, >, etc.), or changing the events to another event type.

The following event resources are available in 800 Mb/s state mode:

- Patterns / Ranges:
  4 patterns, or 2 ranges, or 1 range and 2 patterns on each pod
  - Non-split patterns may use operations: =, ≠, <, ≤, >, ≥, In range, Not in range
  - Split patterns may only use operations: =, ≠
  - Patterns on labels with re-ordered bits may only use operations: =, ≠ (same as 200 and 400 Mb/s modes in all VisiTrigger analysis modules)
- 4 Flags:
  - Flag events from other modules can be checked as set or clear.
  - Flags cannot be set or cleared.
- Arm in from intermodule bus
  - Received from another module in the system or from Port In on the 16700 Series logic analysis system main frame.

Restrictions on the way event resources can be used and combined vary with the type of trigger function selected.

The *Find Pattern* and *Find n Patterns in Immediate Sequence* trigger functions can use all of the available pattern events in any sequence level. However, some combinations (ANDing and ORing) of pattern events, while not exceeding the maximum number of pattern resources, can use too many combiner resources. Any combination of three of the flag or arm in events can also be combined with the pattern events in each sequence level.

When using the *Find n Patterns in Eventual Sequence* trigger functions, only non-split labels may be used and only one pattern event per sequence level may be used. The interface allows insertion of other events in each sequence level to allow ANDing or ORing of the pattern event with other event types (flags or arm in) or replacement of the pattern event with another event type. However, if more than one pattern event is specified, the trigger compiler will be unable to compile it.

Pattern events used in the Default Storing Control count against the number of available resources. Example: Find 4 patterns on a single pod.

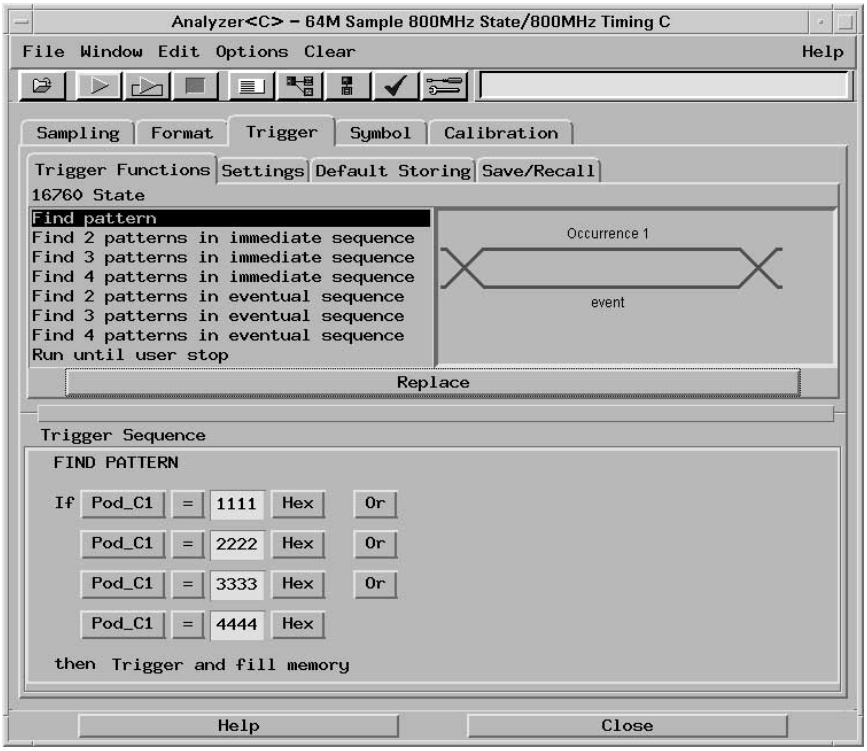


Figure 2. Find 4 patterns on a single pod.

A maximum of 4 patterns on a pod may be specified. These may be ANDed or ORed.

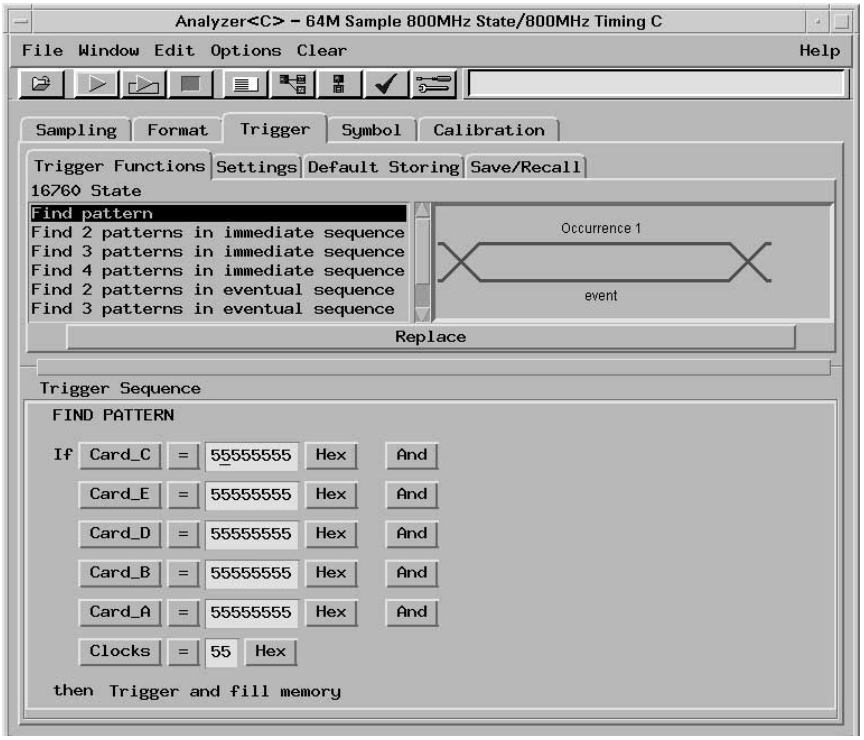


Figure 3. Find a single pattern across the entire analyzer in a 5-card set.

A pattern across all bits of the logic analyzer may be defined by ANDing together multiple labels. The example in figure 3, shows split patterns being used to define a pattern that spans all channels of the logic analyzer.

Although it may at first appear that we have violated the rule of only four pattern resources, notice that any one pod contains only a single pattern resource.

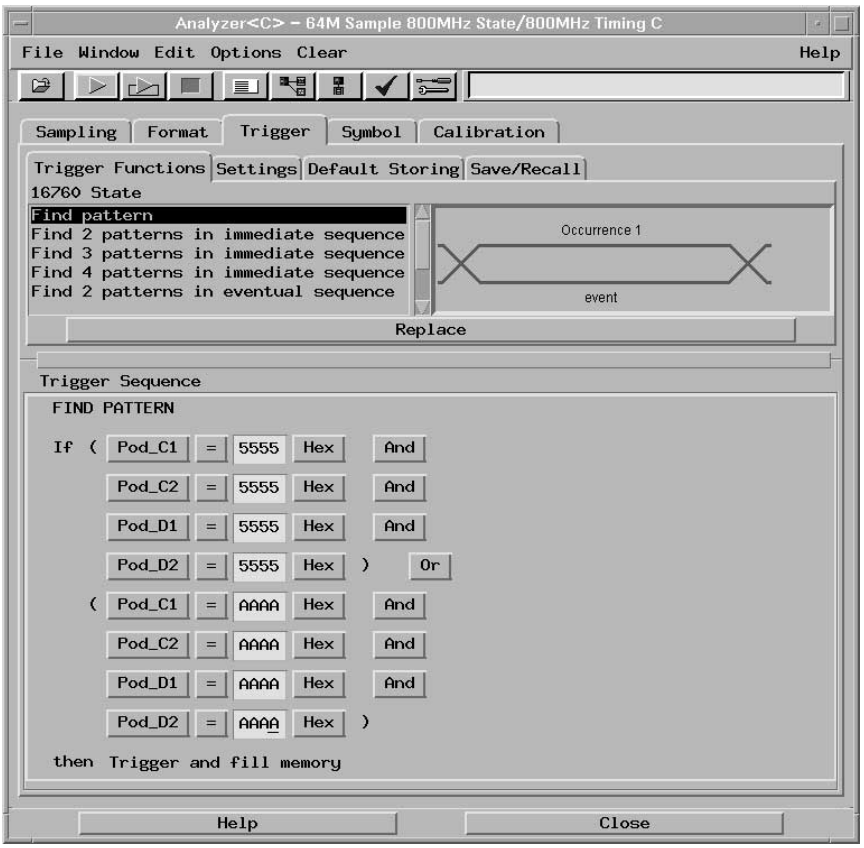


Figure 4. A combination that uses too many combiner resources

This example figure 4, **does not compile**. Even though the example uses only two pattern resources in each pod pair, the way the resources are combined requires more combiner resources than are available in the hardware.

Combiner resources are allocated as needed by the trigger compiler. The way they are allocated is complicated and difficult to explain. There are not any simple rules to use to know when an expression will be too complex to compile, except to say the mixing of AND and OR operations within the same sequence level greatly increases the complexity of the expression to be implemented in hardware.

Parentheses are added automatically to show the order of evaluation when AND / OR functions are mixed. The trigger functions do not, however, allow arbitrary grouping of the AND / OR operation with parentheses by the user.

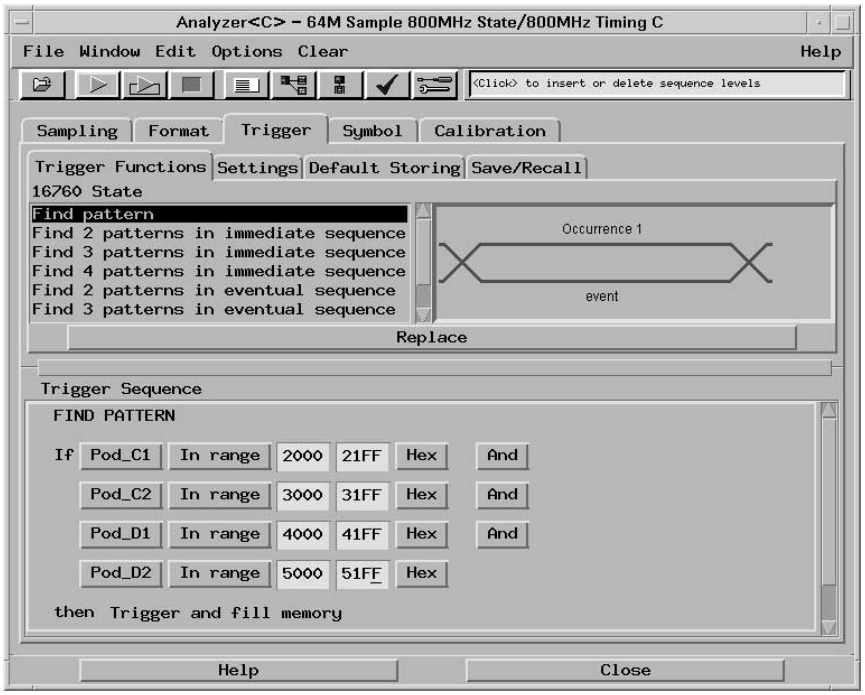


Figure 5. A Multiple ranges

Ranging may be specified only on non-split labels.

This example figure 5, shows four ranges on different labels ANDed together. This trigger specification is compilable because each range is specified on a different pod; each pod has only one pattern range event.

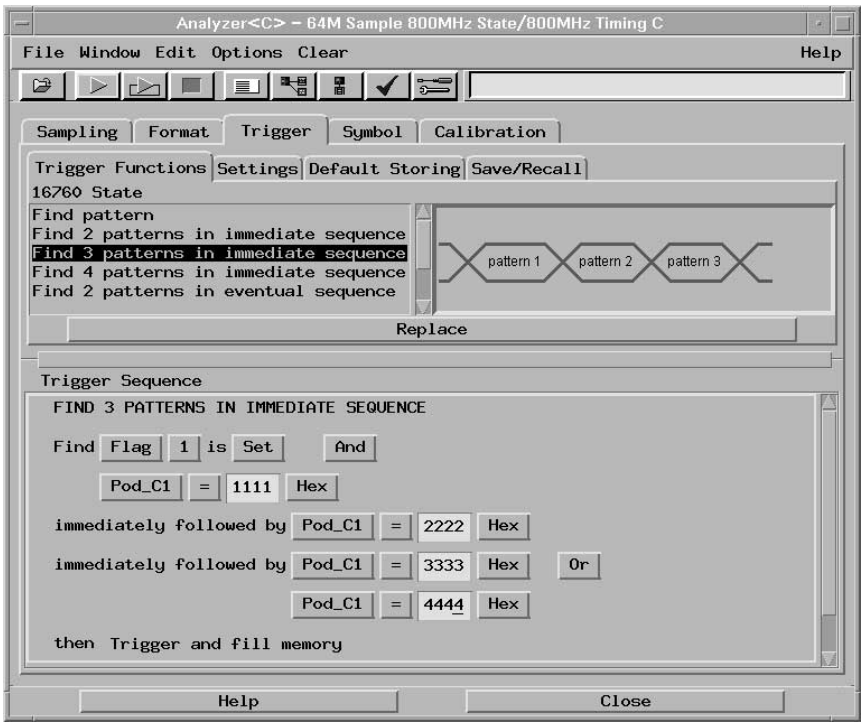


Figure 6. Using 4 patterns in 3 sequence levels.

The Find 3 patterns in immediate sequence trigger function finds patterns (or other events) that occur consecutively, with no other intervening states. If an intervening state does occur, the sequence resets back to looking for the first pattern (or event).

This example figure 6, uses all 4 available pattern resources in pod C1 in 3 sequence levels. The last sequence level uses two pattern resources.

The example also demonstrates using a flag event. The sequence will first wait for the flag to be set true (presumably by another analyzer module in the system) AND Pod_C1 to be 1111h, immediately followed by 2222, then 3333 OR 4444.

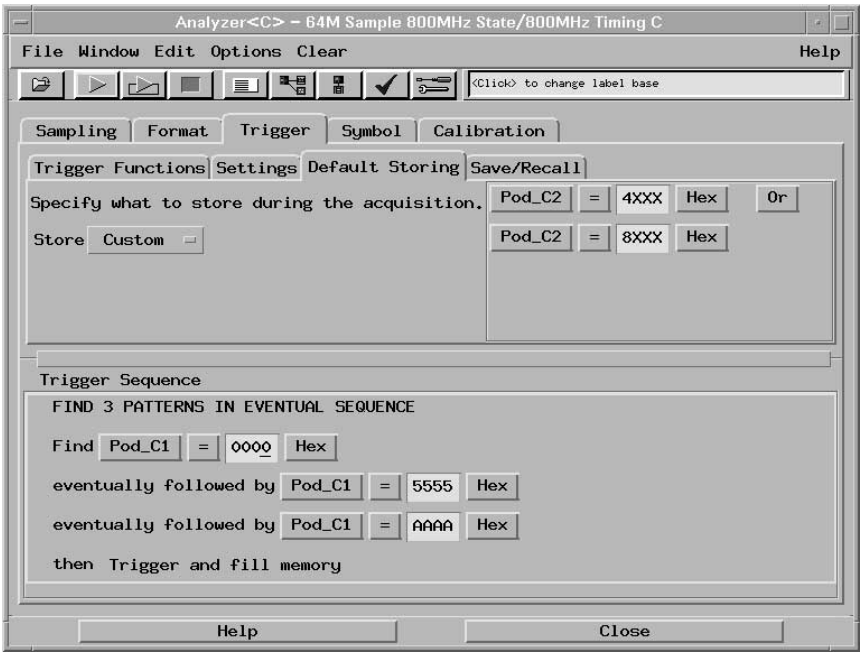


Figure 7. Find patterns in eventual sequence with custom store qualification

The Find n Patterns in Eventual Sequence trigger functions allow finding a sequence of patterns that do not occur consecutively. In the example above, any number of other states can occur between Pod_C1 = 0000 and Pod_C1 = 5555, likewise between the state 5555 and AAAA. Once the desired pattern has been found, the sequencer steps to the next sequence level, but never resets back to looking for the first pattern (nor is there any way to program it to do so).

This example figure 7, also shows usage of the Default Storing control. Default storing is used to conserve trace memory by allowing control of the data to be stored. In the example, only data matching the patterns 4XXX or 8XXX on label Pod_C2 will be stored.

In the example three pattern resources are used in pod C1 and two pattern resources in pod C2 (for default storing).

1250 Mb/s Half Channel State Mode

The 1250 Mb/s half channel state mode provides state speed data rates up to 1250 Mb/s. Only 16 data channel per module, however, may be used. Fewer trigger functions are available and the flexibility of the available trigger functions is less than in the 800 Mb/s state mode.

Half Channel State Format Menu

In half channel state mode, only the even data channels on each probe are used. This is reflected in the format menu as shown in figure 8. In half channel mode, the clock bits associated with each pod cannot be used as data bits.

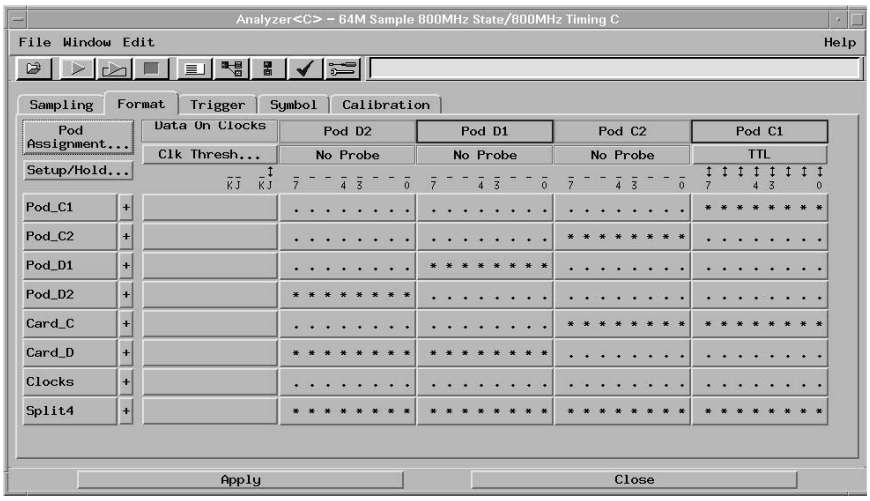


Figure 8. In half channel state mode, it takes a minimum of 4 pods (at 8 bits per pod) to fill a 32 bit split label as shown by label 'Split4'.

1250 Mb/s State Mode Trigger Functions

In 1250 Mb/s state mode the following four trigger functions are available:

- Find pattern
- Find 2 patterns in immediate sequence
- Find 2 patterns in eventual sequence
- Run until user stop

Only one of these functions may be selected at any one time; sequencing of the functions is not allowed.

Each of the functions allow modifications such as ANDing or ORing multiple events, changing the operations on the pattern (=, ≠, <, >, etc.), or changing the events to another event type.

The following event resources are available in 1250 Mb/s state mode:

- Patterns / Ranges:
  3 patterns, or 1 range and 1 pattern on each pod
  - A split pattern uses 1 pattern from each pod in the module (not just the pods with assigned bits).
  - Non-split patterns may use operations: =, ≠, <, ≤, >, ≥, In range, Not in range
  - Split patterns may only use operations: =, ≠
  - Patterns on labels with re-ordered bits may only use operations: =, ≠ (same as 200 and 400 Mb/s modes in all VisiTrigger analyzers)
- 4 Flags:
  - Flag events from other modules can be checked as set or clear.
  - Flags cannot be set or cleared.
- Arm in from intermodule bus
  - Received from another module in the system or the Port In on the 16700 system main frame.

Restrictions on the ways event resources can be used and combined vary with the type of trigger function selected.

The *Find Pattern* and *Find 2 Patterns in Immediate Sequence* trigger functions can use all of the available pattern events in any sequence level. However, some combinations (ANDing and ORing) of pattern events, while not exceeding the maximum number of pattern resources, can use too many combiner resources. Any combination of three of the flag or arm in events can also be combined with the pattern events in each sequence level.

When using the *Find 2 Patterns in Eventual Sequence* trigger function, only non-split labels may be used and only one pattern event per sequence level may be used. The interface allows insertion of other events in each sequence level to allow ANDing or ORing of the pattern event with other event types (flags or arm in) or replacement of the pattern event with another event type. However, if more than one pattern event is specified the trigger compiler will be unable to compile it.

Pattern events used in the Default Storing Control count against the number of available resources.

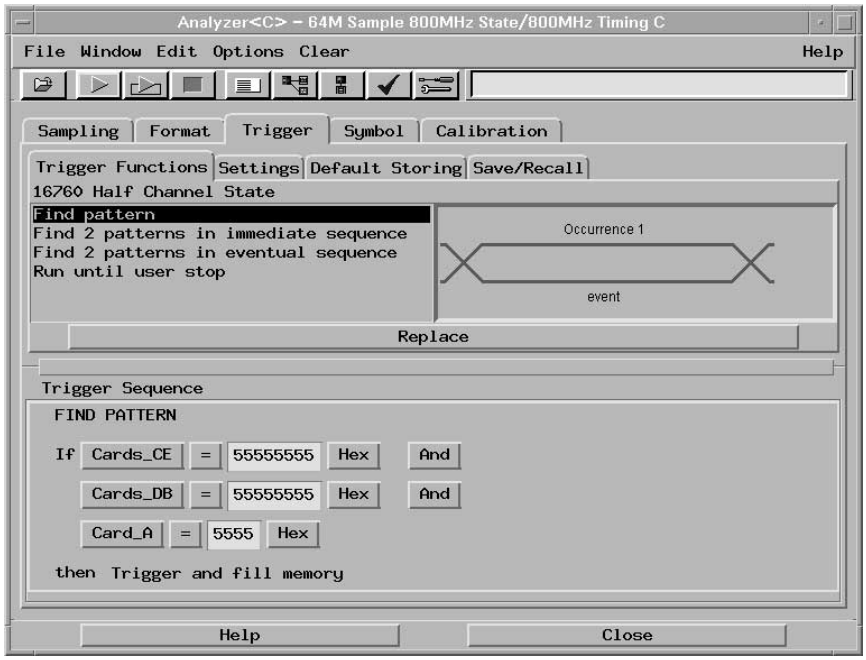


Figure 9. An analyzer-wide pattern on a 5-card module using 32-bit split labels

This example figure 9, uses split labels to implement a pattern across a 5-card module. In half channel mode, two cards (four pods) can be spanned by a single 32-bit label.

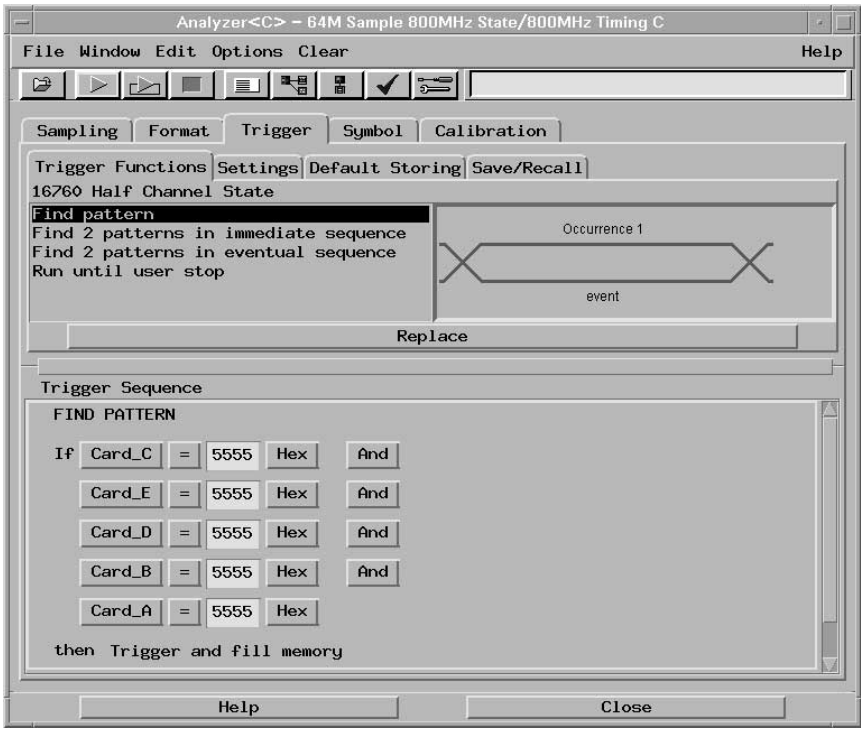


Figure 10. An analyzer-wide pattern on a 5-card module using 16-bit split labels.

This example figure 10, **does not compile.** Although this example implements the same pattern as the previous example, it uses more split pattern labels (five) to implement the function than does the previous example (which used three). Since a split pattern uses a pattern resource from each pod, the maximum of three pattern resources per pod is exceeded.

This causes a problem for the compiler because a split label uses a pattern resource in all pods in the module, not just the ones with bits assigned. The complexity of a split label for the compiler is essentially the same whether the label spans only two pods or multiple pods.

The same function that fails to compile in this example can be implemented by using 32-bit split labels (as shown in the previous example) or by using non-split labels (one per pod).

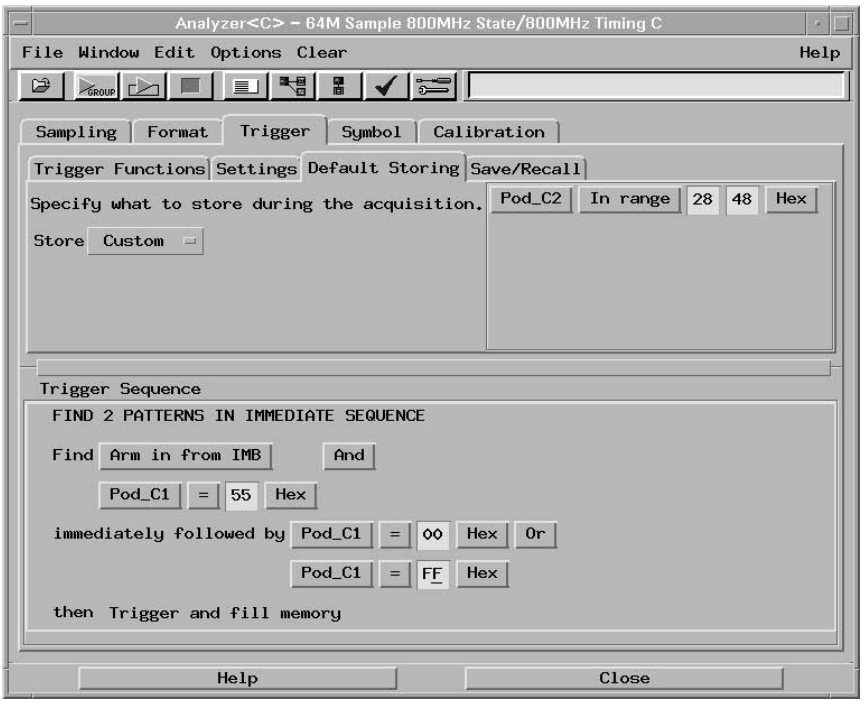


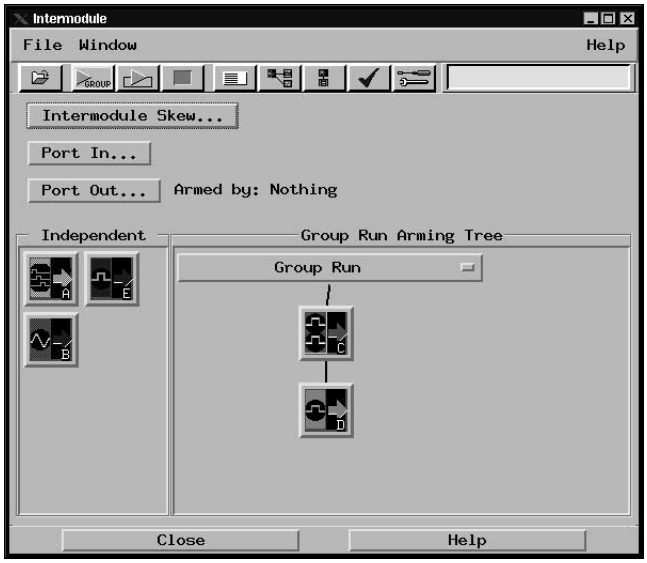
Figure 11. An Arm in from IMB in a Find 2 patterns in immediate sequence trigger function

This example figure 11, uses the *Arm in from IMB* event within a *Find 2 patterns in immediate sequence* trigger function. The function will wait for the arm in event to be true AND the pattern 55 on Pod_C1, immediately followed by the pattern 00 OR the pattern FF on Pod_C1.

The Default Store control has also been set to store into trace memory only samples that fall in the range 28-48 on label Pod_C2.

Note that the Arm in from IMB event is only made available as a selection when the analyzer has been configured in the Intermodule setup dialog to be armed from another module or port in.

Figure 11, shows the 16760A logic analyzer (module D) in the arming tree armed from a 16717A logic analyzer (module C). Figure 12 shows the 16760A logic analyzer armed directly from Port in.



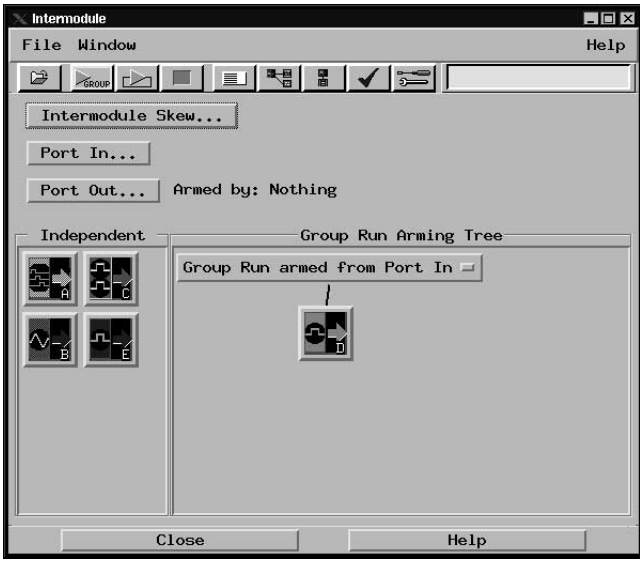


Figure 12 and 13. An IMB setups when using Arm in from IMB event

External Triggering Strategies

Using the Arm in from the intermodule bus event allows the analyzer to be triggered from the IMB trigger-in signal. The IMB trigger-in signal can be generated by the trigger output of another module in the main frame, or from an external signal applied to the Port In BNC connector on the 16700 Series main frame. This capability can be exploited in a number of ways to trigger the logic analyzer at the desired time.

Strategies for using external trigger sources for triggering the logic analyzer include:

- Trigger the logic analyzer from another module or instrument
- Use error detection outputs, etc. provided by the target system
- Modify the software to provide a unique trigger pattern
- Add trigger generators in the target system FPGAs and/or ASICs

Deep memory is an asset when a precise trigger cannot easily be configured, but a trigger that will cause the analyzer to trigger in the near vicinity of the desired location can be generated. In 800 Mb/s state mode, the 16760A provides 64M samples of trace memory, 128M samples in 1250 Mb/s state mode.

Trigger from Another Module or Instrument

Often the trigger position of the analyzer can be determined by probing other (slower speed) buses in the system. This allows for the 16760A or another analyzer to run in a slower speed state mode that provides more complex triggering capabilities. For example, an error on a high-speed local memory access bus may cause a corresponding error on an adjacent (slower speed) PCI bus. In this case the 16760A (or slower-speed logic analyzer module) can probe the slower bus in a state mode that allows more complex triggering to recognize the trigger condition, then trigger a 16760A running in a higher speed state mode with an IMB trigger-in. (Refer to figures 10,11,12).

Trigger on error detection outputs, etc. from the target system

Many target systems provide hardware error or exception signals that may be used for triggering the logic analyzer. Examples would include interrupt signals generated by an error, write strobes generated to an I/O port, dedicated memory location on error, etc.

Modify the target software to provide a unique trigger pattern

Some target systems have the flexibility to modify the software to generate a signal or a unique, identifiable pattern when an error condition occurs, which the logic analyzer can trigger on. Even if the software takes some time to determine an error occurred and then create the trigger condition, deep memory makes it possible to still capture the desired information in trace memory by setting the trigger position near the end of memory.

Add trigger generators in the target system FPGAs and/or ASICs

Some target systems have the flexibility to modify FPGA programming to provide triggering resources. This can be as simple as routing out an internal error signal to an external pin that can be monitored by the analyzer, or a more complex modification to recognize pattern sequences on internal data paths or calculate error such as CRC errors, etc. With pre-planning, ASIC designs can include circuits to be used to provide triggering resources.

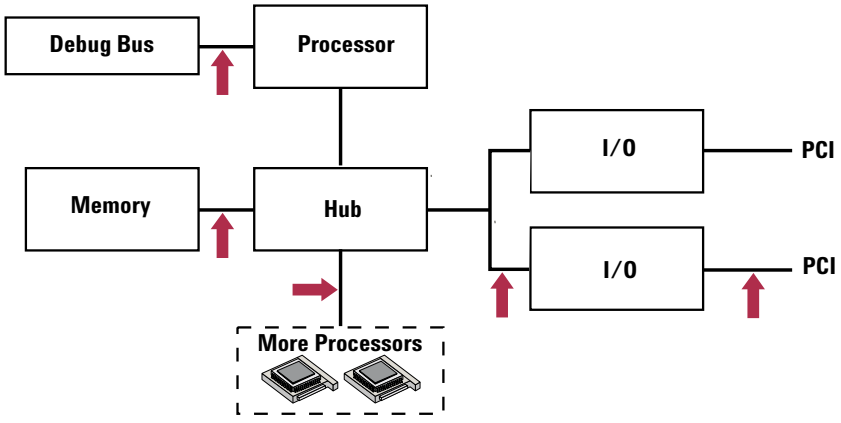


Figure 14. A computer system, Arrows indicate logic analyzer measurement points.

Figure 14 shows one processor of a possible multi-processor computing system. The arrows indicate high-speed buses that may be desirable to probe for debugging hardware, software, and HW/SW integration problems.

The 16760A logic analyer internal triggering capabilities can be used to provide triggering on patterns and simple sequences of patterns on these buses.

The debug buses on many processors can be configured to provide a signal or simple pattern to trigger on.

Errors that can also be detected on one of the PCI buses can be probed with a slower speed logic analyzer with more sophisticated triggering capability. The analyzers probing the high-speed buses can then be triggered by the trigger-out from the PCI bus analyzer.

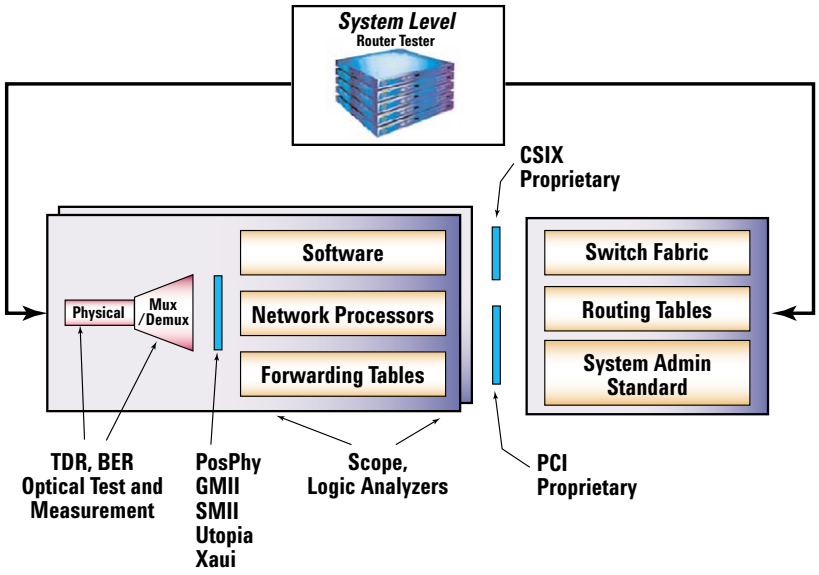


Figure 15. Router system triggering

Figure 15 shows a block diagram of a router system connected to a router test system.

Test systems such as the Agilent RouterTester can be used to stimulate the system with a known pattern at the router inputs and verify that expected data appears at the output. However, when errors do occur these test system provide no visibility on internal buses of the router for debugging and trouble shooting. A logic analyzer may be used to provide access to these internal buses.

Outputs from the test system (on error or at a specified location) may be used to trigger the logic analyzer. This strategy works well with the 16760A triggered from Port In.

As with the computer system example, an analyzer probing the higher speed buses can also be cross-triggered from an analyzer running on one of the slower speed internal buses such as a PCI bus, etc.

200 Mb/s and 400 Mb/s State Modes

Resource	200 Mb/s State	400 Mb/s State
Patterns	16 patterns evaluated as =, ≠, <, >, ≤, ≥	8 patterns evaluated as =, ≠, <, >, ≤, ≥
Ranges	15 ranges evaluated as in range, not in range	4 ranges evaluated as in range, not in range
Timers	2 timers per module	None
Counters	2 global counters, 1 occurrence counter per sequence level	2 occurrence counters
Flags	4 flags evaluated as set, clear	4 flags evaluated as set, clear
Arm in	Arm in from another module on the IMB or from Port In	Arm in from another module on the IMB or from Port In

Table 1. Available trigger resources in the 200 Mb/s and 400 Mb/s modes.

200 Mb/s State	400 Mb/s State
Go to <arbitrary sequence level>	Go to next
Trigger and go to <arbitrary sequence level>	Trigger and fill memory
Trigger and fill memory	
Store/don't store sample	
Turn default storing on/off	
Timer start/stop/pause/resume	
Global counter increment/reset	
Occurrence counter reset	
Flag set/clear	

Table 2. Available trigger actions in the 200 Mb/s and 400 Mb/s modes.

Just as in the 800 and 1250 Mb/s modes, the VisiTrigger interface does not enforce a strict set of limits to guarantee that the specified measurement can be compiled into hardware. Here are some hints in case you encounter the following error messages when the compiler attempts to compile the trigger condition you have specified:

Error message: "Branch expression is too complex"

The "Branch expression is too complex" message means that the event list expression for the indicated branch contains more event terms to logically combine than the hardware is capable of combining on a single branch.

Other branches in the sequence may also be too complex. The trigger sequence compiler stops compiling at the first convenient place after it encounters a fatal error.

Because the trigger sequence compiler tries to optimize the event list expression to best fit the capabilities of the hardware, a precise description of the event list limits cannot be easily enumerated, but some general guidelines for all acquisition modes and some specific suggestions for particular modes are listed on the following page.

General Guidelines

Labels that span multiple pods (split labels) greatly increase the compiled hardware expression complexity, compared with labels that are entirely contained within a single pod.

Whenever possible try to arrange the probing such that labels do not span pods. This is the single most effective way to reduce the complexity required to implement the event list expression.

NOTE:
For labels that do span pods the complexity can be reduced to the same as that of the non-split label case if all bits in the label on all but one pod pair can be set to Xs in the event list expression for the measurement.

For example, if label ADDR has it's 16 most significant bits on pod A2 and 16 least significant bits on pod A1 (spanning pods A2 and A1), the complexity of the compiled expression will be reduced if all 16 MSBs or all 16 LSBs are set to Xs in the pattern event.

Inequality compares (<,≤,>,≥) of split labels increase the expression complexity compared to equality (=,≠) compares of split labels. There is no difference in complexity for non-split labels.

Ranges are implemented as two inequality compares, which doubles the required complexity for non-split labels but increases the complexity to an even greater extent for ranges on split patterns.

Equivalent event list expressions compile to a much greater hardware complexity in 400 Mb/s state mode than in 200 Mb/s state mode. This is due to the way the hardware implements 400 Mb/s state mode. The hardware parallelizes the data in 400 Mb/s mode to allow the internal sequencer to run at ≤ 200 MHz. This requires the trigger compiler to allocate additional sequence levels, branches, and pattern resources and combine them in complex expressions to "de-parallelize" the trigger expression. Using split labels in 400 Mb/s state mode further compounds the complexity of these compiler generated expressions.

The trigger compiler first expands all expression lists to sum-of-products form [e.g. A(B+C) is expanded to AB+AC]. The trigger compiler then does rudimentary Boolean reduction on the expanded expression. However, the compiler makes some trade-offs between complete reduction and compilation speed. Manually expanding and reducing a complex expression may help the trigger compiler to better fit the expression into the hardware resources.

Specific Guidelines - 200 Mb/s state and all timing Modes

Cannot OR more than 16 non-split pattern events if the pattern events are all on the same pod pair.

Cannot OR more than 4 non-split pattern events if each pattern event is on a different pod pair. You can, however, OR 4 patterns together on each of 4 different pods to make a total of 16 patterns OR'd across 4 pods.

Cannot AND more than 16 non-split pattern events if the pattern events are all on the same pod.

Can AND up to 160 non-split pattern events if the pattern events are evenly distributed across all 5 pods on a 5 card set (16 pattern events per pod).

Specific Guidelines - 400 Mb/s State Mode

Cannot AND or OR more than 8 non-split pattern events if the pattern events are all on the same pod.

Cannot OR more than 4 non-split pattern events if each pattern event is on a different pod. You can, however, OR 2 patterns together on each of 4 different pod pairs to make a total of 8 patterns OR'd across 4 pods.

Cannot AND or OR more than 4 non-split ranges if the pattern events are all on the same pod.

Cannot AND or OR more than 2 split equality (=,≠) pattern events.

Cannot specify more than 1 split inequality (<,≤,>,≥) pattern event.

Cannot specify any range on a split label.

In 400 Mb/s state mode, the trigger sequence compiler must combine elements of the trigger events of the previous sequence level and the next sequence with the current sequence level, therefore increasing the total complexity of the current level. A sequence level that may compile fine when it's the only level in the sequence, may be too complex to compile if another level is inserted before or after it.

One possible work around to this limitation is to insert a simple "If anything" sequence level in between two complex levels. The disadvantage to this approach, of course, is that the trigger sequence will miss one state between the two complex sequence levels.

If the following sequence does not compile:

1. If (complex event list) occurs 1 time then Goto Next
2. If (complex event list) occurs 1 time then Trigger and fill memory

This one may:

1. If (complex event list) occurs 1 time then goto next
2. If anything occurs 1 time then Goto Next
3. If (complex event list) occurs 1 time then Trigger and fill memory

In 400 Mb/s state mode, the trigger sequence compiler must always add some additional complexity to the compiled expression for the first sequence level that is not needed in subsequent sequence levels. Additional complexity is also required in the first sequence level for the following 2 conditions:

1. When using the "*Find pattern1, or reset on pattern*2" trigger function in 400 Mb/s state mode, the event list of the first sequence level must be combined with reset branch of each subsequent sequence level by the trigger compiler in order to evaluate the parallelized samples.
2. When using double edge clocking mode (rising and falling edges) in 400 Mb/s state mode, an additional pattern resource is allocated and combined with the event list in the first sequence level by the trigger compiler to prevent triggering on an initial garbage state.

Inserting an "If anything" state as the first state can simplify the complexity of the compiled event list in the first sequence level and subsequent "If/Else" sequence levels. The disadvantage is that the sequence will then miss the first state after the reset condition is met in an "If/Else" sequence level.

If the following sequence does not compile:

1 If (complex event list) occurs 1 time then Goto Next
2 If (complex event list) then Trigger and fill memory Else if (complex event list) then Goto 1

This one may:

1. If anything occurs 1 time then Goto Next
2. If (complex event list) occurs 1 time then Goto Next
3. If (complex event list) then Trigger and fill memory Else if (complex event list) then Goto 1

Error message: “Trigger Specification is too complex”

The "Trigger Specification is too complex" message means that the trigger sequence contains more unique event list expressions than can be allocated to the available combiner resources in the analyzer hardware.

The analyzer has a maximum limit of 16 event list combiner resources. Each unique event list expression requires the use of at least one of these combiner resources. A complex event list may require more than one combiner resource.

The message does not mean that any single event list expression was too complex to combine (see “Branch expression is too complex”), but that the overall number of unique branch expressions specified has exceeded the limit of 16.

In order to compile and run, the total number of unique event list expressions must be reduced to 16 and the complexity of some of the expressions may also have to be reduced.

Branch expressions that are identical (and simple enough to be combined by a single combiner resource - see below) share the same combiner resource. Reusing identical event list equations where possible will optimize the use of combiner resources.

Combiner resource allocation guidelines

Labels that span multiple pods (split labels) increase the number of required combiner resources as compared with labels that are entirely contained within a single pod.

Whenever possible try to arrange the probing such that labels do not span pods. This is the single most effective way to reduce the number of required combiner resources.

NOTE: For labels that do span pods, the complexity can be reduced to the same as that of the non-split label case, if all bits in the label on all but one pod can be set to Xs in the event list expression for the measurement.

For example, if label ADDR has it's 16 most significant bits on pod A2 and 16 least significant bits on pod A1 (spanning pods A2 and A1), the complexity of the compiled expression will be reduced if all 16 MSBs or all 16 LSBs are set to Xs in the pattern event.

Event lists with up to 4 unique pattern events can be combined in any combination of ANDs and ORs by a single combiner resource if all of the pattern labels are non-split and contained on the same pod.

Combining more than 4 labels on the same pod will require another combiner resource.

Non-split label pattern events from different pod pairs that are OR'd together require an additional combiner resource for each additional pod included in the event list. (ANDing on non-split pattterns from different pod does not increase the required number of combiner resources).

An inequality compare (<,≤,>,≥) with a split label pattern event requires 2 combiner resources.

A range on a split label pattern event requires 4 combiner resources.

The event list in the custom store qualification dialog also allocates combiner resources from the same pool of 16 resources. If the store qualification event list equation is the same as one of the branch event list equations in the trigger sequence, the combiner resource will be shared. A unique store qualification event list requires the allocation of 1 (or more) of the combiner resources.

400 Mb/s state mode requires many more combiner resources to implement the same trigger sequence as compared to 200 Mb/s state mode and all timing modes. Refer to the discussion of complexity in the "Branch expression is too complex" topic above.

Agilent Technologies' Test and Measurement Support, Services, and Assistance
Agilent Technologies aims to maximize the value you receive, while minimizing your risk and problems. We strive to ensure that you get the test and measurement capabilities you paid for and obtain the support you need. Our extensive support resources and services can help you choose the right Agilent products for your applications and apply them successfully. Every instrument and system we sell has a global warranty. Support is available for at least five years beyond the production life of the product. Two concepts underlie Agilent's overall support policy: "Our Promise" and "Your Advantage."

Our Promise
Our Promise means your Agilent test and measurement equipment will meet its advertised performance and functionality. When you are choosing new equipment, we will help you with product information, including realistic performance specifications and practical recommendations from experienced test engineers. When you use Agilent equipment, we can verify that it works properly, help with product operation, and provide basic measurement assistance for the use of specified capabilities, at no extra cost upon request. Many self-help tools are available.

Your Advantage
Your Advantage means that Agilent offers a wide range of additional expert test and measurement services, which you can purchase according to your unique technical and business needs. Solve problems efficiently and gain a competitive edge by contracting with us for calibration, extra-cost upgrades, out-of-warranty repairs, and onsite education and training, as well as design, system integration, project management, and other professional engineering services. Experienced Agilent engineers and technicians worldwide can help you maximize your productivity, optimize the return on investment of

Agilent Email Updates

www.agilent.com/find/emailupdates
Get the latest information on the products and applications you select.

Agilent Direct

www.agilent.com/find/agilentdirect
Quickly choose and use your test equipment solutions with confidence.

Agilent Technologies Warranty

Agilent hardware products are warranted against defects in materials and workmanship for a period of one year from date of shipment. Some newly manufactured Agilent products may contain remanufactured parts, which are equivalent to new in performance. If you send us a notice of such defects during the warranty period, we will either repair or replace hardware products that prove to be defective. Agilent software and firmware products that are designated by Agilent for use with a hardware product are warranted for a period of one year from date of shipment to execute their programming instructions when properly installed. If you send us notice of defects in materials or workmanship during the warranty period, we will repair or replace these products, so long as the defect does not result from buyer supplied hardware or interfacing. The warranty period is controlled by the warranty statement included with the product and begins on the date of shipment.

For more information on Agilent Technologies products, applications or services, please contact your local Agilent office.

The complete listing is available at:
www.agilent.com/find/contactus

Phone or Fax
United States:
(tel) 800 829 4444
(fax) 800 829 4433

Canada:
(tel) 877 894 4414
(fax) 905 282 6495

China:
(tel) 800 810 0189
(fax) 800 820 2816

Japan:
(tel) (81) 426 56 7832
(fax) (81) 426 56 7840

Korea:
(tel) (080) 769 0800
(fax) (080) 769 0900

Latin America:
headquarters:
(tel) (305) 269 7500

Taiwan:
(tel) 0800 047 866
(fax) 0800 286 331

Other Asia Pacific Countries:
(tel) (65) 6375 8100
(fax) (65) 6755 0042
Email: tm_ap@agilent.com

Product specifications and descriptions in this document subject to change without notice.

Printed in USA June 28, 2004

5988-2994EN

Agilent Technologies