Булевы соотношения

Boolean_expression – это набор логических операторов, операторов отношений и их операндов, которые вычисляют булевый результат. В зависимости от оператора, операнд может обращаться к состоянию триггера, счётчику и регистру, или числовому значению. Внутри выражения могут использоваться круглые скобки для группирования набора операндов.

Логические операторы используют некоторые булевы выражения как операнды. Поддерживаемые логические операторы приведены в таблице 14-6.

Таблица 14-6. Логические операторы

Оператор	Описание	Синтаксис
!	оператор НЕ	!expr1
&&	оператор И	expr1 && expr2
||	оператор ИЛИ	expr1 || expr2

Операторы отношения выполняются на счётчиках или флагах статуса. Сравниваемое значение – правый оператор – должен иметь числовое значение. Поддерживаемые операторы отношения приведены в таблице 14-7.

Table 14–7. Relational Operators

Operator	Description	Syntax *(Note 1) (2)*
>	Greater than	<identifier> > <numerical_value>
>=	Greater than or Equal to	<identifier> >= <numerical_value>
==	Equals	<identifier> == <numerical_value>
!=	Does not equal	<identifier> != <numerical_value>
<=	Less than or equal to	<identifier> <= <numerical_value>
<	Less than	<identifier> < <numerical_value>

Примечания к таблице 14-7:
(1) <identifier> показывает счётчик или флаг статуса;
(2) <numerical_value> - целочисленные значения.

Список действий

Action_list – это список действий, которые могут выполняться, когда наступает состояние и условие также выполняется. Если определяются более одного действия, необходимо ограничить их с помощью *begin* и *end*. Действия делятся на действия манипуляции с ресурсами, действия контроля буфера и действия переходов состояний. Каждое действие заканчивается точкой с запятой (;).

Действия манипуляции с ресурсами

Ресурсы, используемые для описания процесса триггера, состоят из счётчиков и флагов статуса. В таблице 14-8 показаны описание и синтаксис каждого действия.

Таблица 14-8. Действия манипуляции с ресурсами (часть 1 из 2)

Действия	Описание	Синтаксис
инкремент	Увеличивает ресурс счётчика на 1	increment <*counter_identifier*>;
декремент	Уменьшает ресурс счётчика на 1	decrement <*counter_identifier*>;

Таблица 14-8. Действия манипуляции с ресурсами (часть 2 из 2)

Действия	Описание	Синтаксис
сброс	Сбрасывае ресурс счётчика в начальное значение	`reset <counter_identifier>;`
установка	Устанавливает флаг статуса в 1	`set <register_flag_identifier>;`
снятие	Устанавливает флаг статуса в 0	`clear <register_flag_identifier>;`

Действия контроля буфера

Действия контроля буфера определяют действия по контролю над буфером захвата. В таблице 14-9 показано описание и синтаксис каждого действия.

Таблица 14-9. Действия контроля буфера

Действия	Описание	Синтаксис
`trigger`	Останавливает захват для текущего буфера и заканчивает анализ. Эта команда необходима для определения каждого процесса.	`trigger <post-fill_count>;`
`segment_trigger`	Заканчивает захват для текущего сегмента. Встроенный логический анализатор SignalTap II начинает захват со следующего сегмента, вычисленного этой командой. Если все сегменты заполнены, старейший сегмент затирается последним отсчётом. Захват останавливается, когда вычисляется действие триггера. Это действие не может использоваться для несегментного режима захвата.	`segment_trigger <post-fill_count>;`
`start_store`	Связывает `write_enable` с буфером захвата SignalTap II. Эта команда активна только, если разрешён режим квалификатора базовых состояний памяти.	`start_store`
`stop_store`	Разрывает связь `write_enable` с буфером захвата SignalTap II. Эта команда активна только, если разрешён режим квалификатора базовых состояний памяти.	`stop_store`

Оба действия `trigger` и `segment_trigger` связаны опциональным аргументом счётчика пост-заполнения. Если он предусмотрен, то текущий захват запрашиваемого количества отсчётов обеспечивается счётом пост-заполнения, и он останавливает захват. Если значение пост-счёта не определено, позиция триггера, влияющая на буфер, устанавливается по умолчанию на позицию, определённую на вкладке **Установка.**

В случае с `segment_trigger`, захват в текущем буфере останавливается сразу, если следующее действие защёлкивания происходит в следующем состоянии, не зависимо от того пост-заполнение или не заполнение выполнено в текущем буфере. Оставшийся незаполненный захват пост-счёта в текущем буфере отбраковывается и отображается как затенённые отсчёты в окне данных.

Действие перехода состояния

Действие перехода состояния определяет следующее состояние в процессе контроля настраиваемых состояний. Оно определяется командой `goto`. Синтаксис следующий:

```
goto <state_label>;
```

Использование средства квалификатора памяти базовых состояний

Когда вы выбрали тип *базовое состояние* для квалификатора памяти, действия `start_store` и `stop_store` разрешаются в процессе базовых состояний триггера. Эти команды, которые используются вместе с выражениями процесса базовых состояний триггера, дают вам максимальную гибкость для контроля данных, записываемых в буфер захвата.

Команды `start_store` и `stop_store` применяются только для несегментных буферов.

Команды `start_store` и `stop_store` функционально похожи на состояния *старт* и *стоп*, когда используется режим квалификатора состояний **старт/стоп**. Если квалификация памяти разрешена, команда `start_store` должна выполнятся в SignalTap II для записи данных в буфер захвата. Никакие данные не будут захвачены, пока не выполнится команда `start_store`. Также команда `trigger` должна включаться в описание процесса триггера. Команда `trigger` необходима для полного захвата и отображения результатов на экране диаграмм.

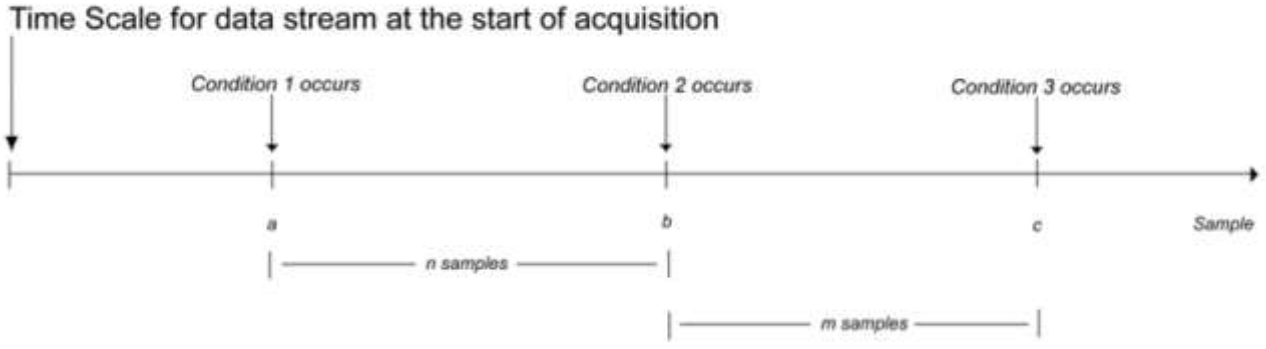
В следующих примерах проиллюстрировано поведение процесса базовых состояний триггера с командами квалификации памяти.

На рисунке 14-34 показан гипотетический сценарий с тремя состояниями триггера, которые случаются в различных точках на временной шкале, после нажатия кнопки анализа. Описание процесса триггера в примере 14-2, которое применяется к сценарию рисунка 14-34, показывает функционирование средства квалификации памяти для процесса базовых состояний триггера.

Example 14–2. Trigger Flow Description 1

```
State 1: ST1:
if ( condition1 )
   start_store;
else if ( condition2 )
   trigger value;
else if ( condition3 )
   stop_store;
```

Figure 14–34. Capture Scenario for Storage Qualification with the State-Based Trigger Flow



В этом примере, встроенный логический анализатор SignalTap II не записывает в буфер захвата, пока не произойдёт Состояние 1, отсчёт *a*. После отсчёта *b* вычисляется команда `trigger value`. Логический анализатор продолжает записывать в буфер до окончания захвата. Процесс захвата определяет команду `stop_store` на отсчёте *c*, через *m* отсчётов после возникновения точки триггера.

Логический анализатор способен остановить захват и отобразить содержимое диаграмм, если он успешно закончился на отсчёте захвата пост-заполнения, перед Состоянием 3. В этом конкретном случае, захват заканчивается, если значение пост-заполнения счета меньше, чем *m*.

Если значения счёта пост-заполнения, определённое в описании процесса триггера, больше чем *m* отсчётов, буфер приостанавливает захват на неопределённое время, при этом не будет повторения Состояния 1 для защёлкивания логического анализатора и начала повторного захвата данных. Встроенный логический анализатор SignalTap II продолжит исполнять команды **stop_store** и **start_store** сразу после исполнения команды **trigger**. Если захват приостановлен, вы можете вручную остановить и форсировать захват триггером, используя кнопку **Стоп анализ**. Вы можете использовать значения счетчика, флаги и диаграмму состояний для того, чтобы отследить исполнение процесса триггера. Значения счётчиков, флагов и текущего состояния обновляется в реальном времени во время захвата данных.

На рисунках 14-35 и 14-36 показан захват данных в реальном времени по сценарию, описанному на рисунке 14-33. На рисунке 14-35 проиллюстрирован сценарий, когда захват данных заканчивается успешно. Здесь используется буфер с глубиной захвата 64, $m = n = 10$, а значение пост-заполнения равно 5. На рисунке 14-36 проиллюстрирован сценарий, когда логический анализатор приостанавливается на неопределённый срок, сразу после совершения состояния триггера, до состояния `stop_store`. Здесь используется буфер с глубиной захвата 64, $m = n = 10$, а значение пост-заполнения равно 15.

Figure 14–35. Storage Qualification with Post-Fill Count Value Less than *m* (Acquisition successfully completes)

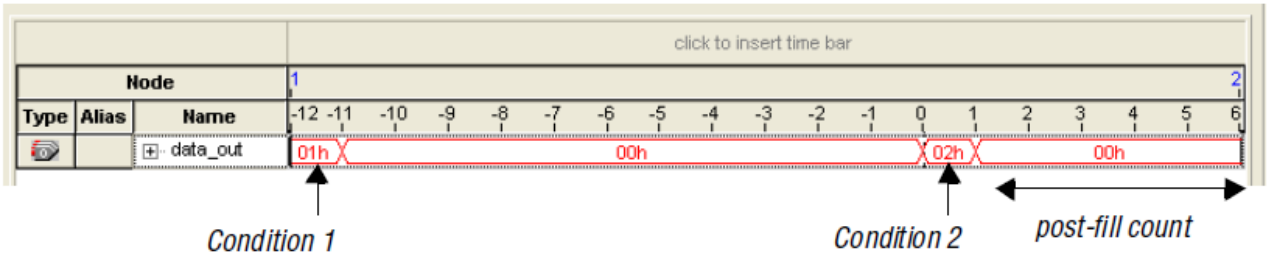
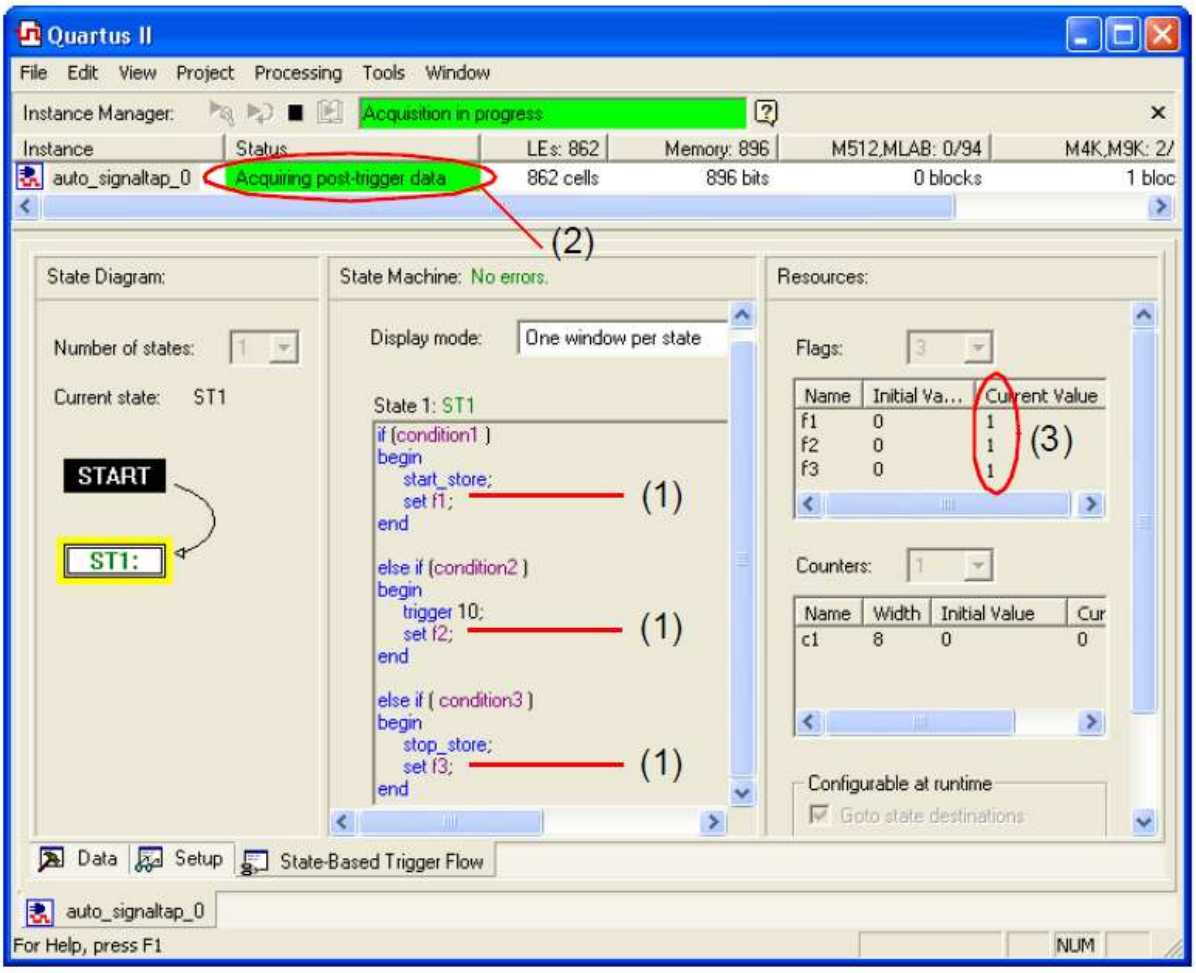
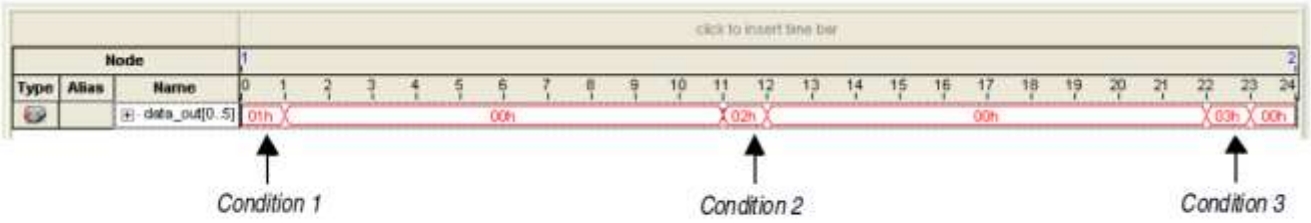


Figure 14–36. Storage Qualification with Post-Fill Count Value Greater than *m* (Acquisition indefinitely paused)



Временные диаграммы после форсирования анализа до остановки



(1) Флаг добавляется к процессу триггера, чтобы помочь отследить исполнение во время запуска.

(2), (3) Панель статуса и поля текущего значения обновляются во время захвата, демонстрируя состояние данных захвата в реальном времени.

Комбинация использования счётчиков, булевых и условных операторов вместе с командами `start_store` и `stop_store` может дать разрешение на уровне тактовых циклов для контроля отсчётов, которые будут записываться в буфер захвата. В примере 14-3 показано описание процесса триггера, где пропускаются три тактовых цикла отсчётов после возникновения состояния 1. На рисунке 14-37 показаны переходы данных в следующие состояния, а на рисунке 14-38 показан захват данных с помощью описания процесса триггера из примера 14-3.

Example 14–3. Trigger Flow Description 2

```
State 1: ST1
start_store
if ( condition1 )
begin
   stop_store;
   goto ST2;
end


State 2: ST2
if (c1 < 3)
   increment c1; //skip three clock cycles; c1 initialized to 0

else if (c1 == 3)
begin
   start_store; //start_store necessary to enable writing to finish
                //acquisition
   trigger;
end
```

Figure 14–37. Continuous Capture of Data Transaction for Example 2

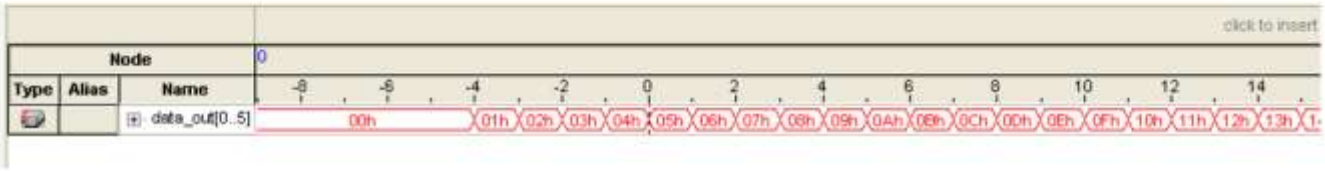


Figure 14–38. Capture of Data Transaction with Trigger Flow Description Applied