

16. Обновление памяти и констант в системе

Введение

Редактор содержимого памяти в системе позволяет вам наблюдать и обновлять память и константы, используя соединение по JTAG порту. В этой главе показано, как использовать Редактор содержимого памяти в системе Quartus II как части вашего FPGA процесса проектирования и отладки.

Проекты FPGA существенно возрастают в плотности и получают достаточно комплексными. Разработчикам и инженерам-отладчикам требуется значительный доступ к проекту, запрограммированному в чипе, для ускорения идентификации, тестирования и конечного выпуска. Обновление в системе памяти и констант, возможное в программе Quartus® II, позволяет иметь доступ к считыванию (записи) памяти и констант в системе FPGA посредством интерфейса JTAG, делая проще тестовые изменения содержимого памяти FPGA, во время функционирования FPGA в конечной системе.

Эта глава состоит из следующих разделов:

- "Поддержка мегафункций чипом" на странице 16-2
- "Использование обновления в системе памяти и констант в вашем проекте" на странице 16-2
- "Создание модифицируемых в системе памяти и констант" на странице 16-3
- "Запуск редактора содержимого памяти в системе" на странице 16-3

Общее представление

Способность читать и обновлять память и константы в запрограммированном чипе позволяет больше проникать внутрь и внешне контролировать ваш проект. Редактор содержимого памяти в системе Quartus II даёт вам доступ к памяти и константам чипа. Когда вы используете его вместе со встроенным логическим анализатором SignalTap® II, эти средства придают вам необходимую наглядность во время отладки вашего проекта в лабораторном макете.

Больше информации о встроенном логическом анализаторе SignalTap® II находится в главе "Отладка проекта с помощью встроенного логического анализатора SignalTap II" в томе 3 "Настольной книги Quartus II".

Способность читать данные из памяти и констант позволяет вам быстро идентифицировать источник проблем. Дополнительно, способность записи позволяет вам обходить функциональные результаты путём записи нужных данных. Например, если бит паритета в вашей памяти некорректный, вы можете использовать редактор содержимого памяти в системе для записи значения корректного бита паритета в вашу RAM, позволяя вашей системе дальнейшее функционирование. А также вы можете записать внутренне некорректное значение бита паритета в вашу RAM, чтобы проверить устойчивость вашего проекта к ошибке.

Программа Quartus II предлагает набор инструментов отладки в чипе. Для просмотра и сравнения всех доступных инструментов в наборе инструментов отладки в чипе программы Quartus II, обратитесь к главе "V. Отладка в системе" в томе 3 "Настольной книги Quartus II".

Поддержка мегафункций чипом

В следующих таблицах приведены чипы и типы памяти и констант, которые свободно поддерживаются программой Quartus II. В таблице 16-1 показаны типы памяти, поддерживаемые менеджером плагина MegaWizard™ и редактором содержимого памяти в системе.

Таблица 16-1. Поддержка менеджера плагина MegaWizard™

Категория установленных плагинов	Название мегафункции
Вентили	LPM_CONSTANT
Компилятор памяти	RAM: 1-PORT, ROM: 1-PORT
Запоминающее устройство	ALTSYNCRAM, LPM_RAM_DQ, LPM_ROM

В таблице 16-2 показана поддержка обновлений в системе памяти и констант для чипов семейств Stratix®, Arria® GX, Cyclone®, APEX™ II и APEX 20K.

Таблица 16-2. Поддержка мегафункций

Мегафункция	Arria GX / Stratix Series			Cyclone Series	APEX II	APEX 20K
	M512 Blocks	M4K Blocks	MegaRAM Blocks			
LPM_CONSTANT	чтение-запись	чтение-запись	чтение-запись	чтение-запись	чтение-запись	чтение-запись
LPM_ROM	запись	чтение-запись	нет	чтение-запись	чтение-запись	запись
LPM_RAM_DQ	нет	чтение-запись	чтение-запись	чтение-запись	чтение-запись	нет (1)
ALTSYNCRAM (ROM)	запись	чтение-запись	нет	чтение-запись	нет	нет
ALTSYNCRAM (Режим однопортовой RAM)	нет	чтение-запись	чтение-запись	чтение-запись	нет	нет

Примечание к таблице 16-2: (1) Только в режиме "только запись" для этой однопортовой RAM. В режиме "только чтение", используйте LPM_ROM вместо LPM_RAM_DQ.

Использование обновления в системе памяти и констант в вашем проекте

Использование средства обновления в системе памяти и констант в вашем проекте подразумевает выполнение следующих шагов:

1. Определите память и константы, к которым вы хотите иметь доступ.
2. Отредактируйте память и константы так, чтобы их можно было модифицировать в реальном времени.
3. Выполните полную компиляцию.
4. Запрограммируйте ваш чип.
5. Запустите редактор содержимого памяти в системе.

Создание модифицируемых в системе памяти и констант

Когда вы определите память или константу для модификации в реальном времени, программа Quartus II заменит её исполнение по умолчанию. Однопортовая RAM конвертируется в двухпортовую RAM, а константы выполняются в регистрах вместо кодовых таблиц (LUT). Эти изменения позволяют модифицировать в реальном времени без изменения функциональности вашего проекта. Список мегафункция для изменений в реальном времени приведён в таблице 16-1.

Чтобы разрешить модифицировать вашу память или константу, выполните следующие шаги:

1. В меню **Инструменты**, кликните **менеджер плагинов MegaWizard**.
2. Если вы создаёте новую мегафункцию, выберите **Создать новую свободную вариацию мегафункции**. Если у вас есть уже существующая мегафункция, выберите **Редактировать существующую свободную вариацию мегафункции**.
3. Внесите изменения в мегафункцию, основываясь на требуемых характеристиках вашего проекта, включите **Разрешить редактор содержимого памяти в системе для сбора и обновления содержимого**, не зависимо от системного таймера, а также типа и значения в текстовом модуле **Идентификатора Элемента**. Эти параметры можно найти на последней странице MegaWizard мегафункций, поддерживающих обновление в системе.

Идентификатор Элемента – это строка из четырёх символов, используемая для различения мегафункции от других элементов памяти и констант в системе.

4. Кликните **Финиш**.
5. В меню **Процессы**, кликните **Старт компиляции**.

Если вы инициализируете память или константу мегафункции, только используя порты и параметры в VHDL или Verilog HDL, то следующим способом добавьте параметр `lpm_hint`:

В коде VHDL добавьте следующее:

```
lpm_hint => "ENABLE_RUNTIME_MOD = YES,  
INSTANCE_NAME = <instantiation name>;
```

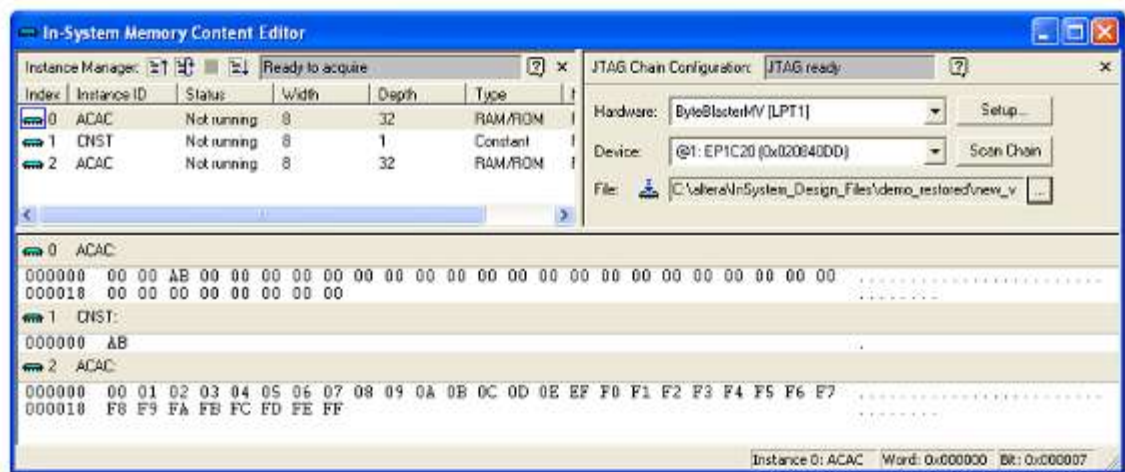
В коде Verilog HDL добавьте следующее:

```
defparam <megafunction instance name>.lpm_hint =  
"ENABLE_RUNTIME_MOD = YES,  
INSTANCE_NAME = <instantiation name>;
```

Запуск редактора содержимого памяти в системе

Редактор содержимого памяти в системе делится на Менеджер элемента, Конфигурацию цепи JTAG и Редактор Нех (рисунок 16-1).

Figure 16–1. In-System Memory Content Editor



Менеджер элемента показывает всю доступную для изменений в реальном времени память и константы в вашем чипе FPGA. Секция конфигурации цепи JTAG позволяет вам программировать ваш FPGA и выбирать чип Altera® в цепи JTAG для обновления.

Для использования редактора содержимого памяти в системе не требуется открытие проекта. Редактор содержимого памяти в системе находит все элементы конфигурируемой в реальном времени памяти и констант путём сканирования цепи JTAG и посыланием запроса определённым чипам, выбранным в секции конфигурации цепи JTAG.

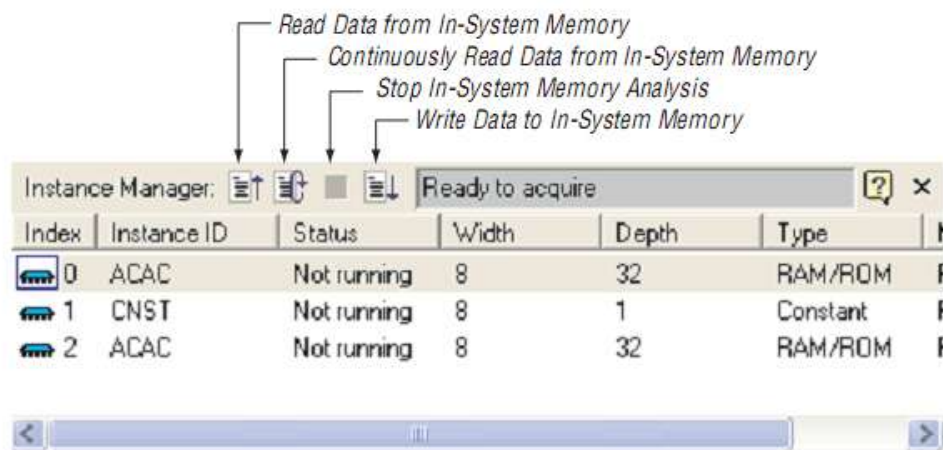
Каждый редактор содержимого памяти в системе имеет доступ к внутренним блокам памяти и констант одного чипа. Если у вас имеется более одного чипа, содержащего элементы конфигурируемой в реальном времени памяти или констант в цепи JTAG, вы можете запустить несколько редакторов содержимого памяти в системе внутри программы Quartus II, чтобы иметь доступ к элементам памяти и констант для каждого чипа.

Менеджер элемента

Сканирование цепи JTAG приводит к обновлению менеджера элемента списком всех элементов конфигурируемых в реальном времени памяти и констант в проекте. Менеджер элемента показывает: индекс, элемент, статус, ширину, глубину и режим каждого элемента в списке.

Вы можете считывать и записывать в системную память, используя менеджер элемента, как это показано на рисунке 16-2.

Figure 16–2. Instance Manager Controls



Следующие кнопки представлены в менеджере элемента:

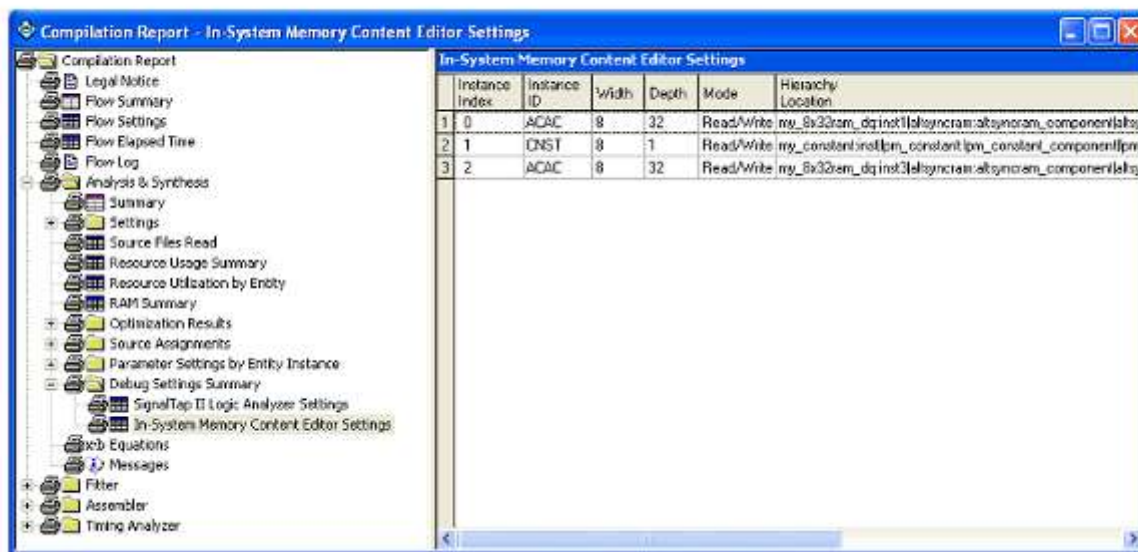
- **Прочитать данные из системной памяти** – читает данные из чипа, не зависимо от системного таймера и выводит в редакторе Hex.
- **Непрерывное чтение из системной памяти** – непрерывное чтение данных асинхронно из чипа и отображение в редакторе Hex.
- **Остановка анализа системной памяти** – останавливает текущую операцию чтения или записи
- **Записать данные в системную память** – асинхронная запись в чип данных, представленных в редакторе Hex.

В дополнении к кнопкам, доступным в менеджере элемента, вы можете также читать и записывать данные выбором команды в меню **Процессы**, или после правого клика в менеджере элемента или редакторе Hex во всплывающем окне.

Статус каждого элемента также показывается рядом с каждым модулем в менеджере элемента. Статус показывает, что, либо модуль **не запущен**, либо **перезагружаются данные**, либо **данные обновляются**. Дисплей состояния даёт полную информацию о статусе редактора.

Программа Quartus II назначает различный номер индекс для каждой системной памяти или константы, чтобы различать модули памяти или функции констант. Посмотрите настройки редактора содержимого памяти в системе в отчёте компиляции для сверки индекса с соответствующими идентификаторами элемента.

Figure 16–3. Compilation Report In-System Memory Content Editor Settings Section



Instance Index	Instance ID	Width	Depth	Mode	Hierarchy Location
1	0	ACAC	8	32	Read/Write nw_8x32ram_dqinst1lsyncram:alsyncram_component1lsyncram
2	1	CONST	8	1	Read/Write m0_constantinst1pm_constant1pm_constant_component1pm
3	2	ACAC	8	32	Read/Write m0_8x32ram_dqinst3lsyncram:alsyncram_component1lsyncram

Редактирование данных в редакторе Нех

Вы можете редактировать данные, прочитанные из ваших элементов системной памяти и констант, отображаемые в Редакторе Нех, вводом значений вручную непосредственно в редакторе, или импортом файлов памяти.

Для изменения данных, отображаемых в редакторе Нех, кликните на место в редакторе и введите или вставьте из буфера обмена новые данные. Новые данные отображаются голубым цветом, показывающим изменённые данные, которые ещё не записаны в FPGA. В меню **Редактировать**, выберите **Значение**, и кликните **Заполнить "0"**, **Заполнить "1"**, **Заполнить "Случайным значением"** или **Обычное заполнение**, для обновления блока данных выбранным блоком данных.

Импорт и экспорт файлов памяти

Редактор содержимого памяти в системе позволяет вам выбрать файл для импорта и экспорта значения данных в и из памяти, для которой разрешено средство обновления в системе. Импорт из файлов данных позволяет вам быстро загружать в память всё изображение. Экспорт в файлы данных позволяет вам сохранять содержимое памяти для дальнейшего использования и анализа.

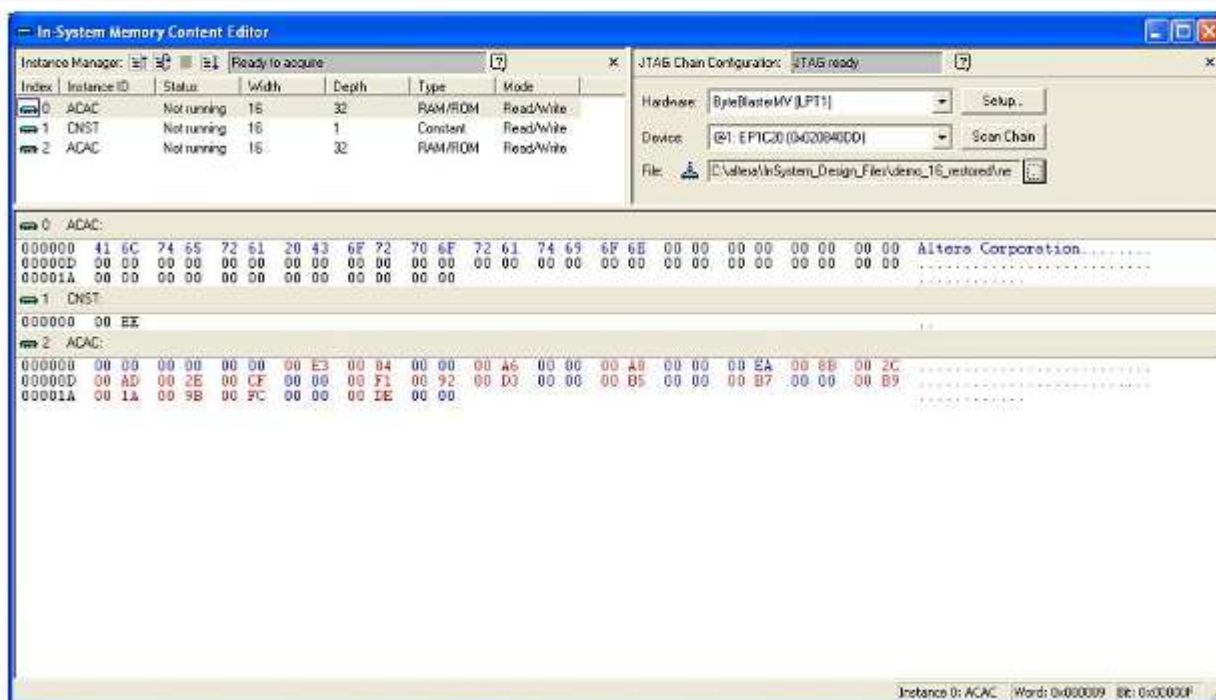
Для импорта файла памяти, используя редактор содержимого памяти в системе, выберите в менеджере элементов память или константу, которую вы хотите использовать. В меню **Редактировать**, кликните **Импорт данных из файла**, и выберите файл данных, который вы хотите загрузить в выбранную память или константу. Вы можете импортировать только файл памяти в формате шестнадцатеричного файла Intel (**.hex**) или в формате файла инициализации памяти (**.mif**).

Аналогично, для экспорта содержимого памяти в файл, используя редактор содержимого памяти в системе, выберите в менеджере элементов память или константу, которую вы хотите использовать. В меню **Редактировать**, кликните **Экспорт данных из файла**, и выберите имя файла, в котором вы хотите сохранить данные. Вы можете экспортировать данные в форматах **.hex**, **.mif**, файла изменения значения дампа Verilog (**.vcd**) или файла инициализации RAM (**.rif**).

Просмотр содержимого памяти и констант

Для каждого модуля системной памяти или константы, редактор Нех выводит данные в виде шестнадцатеричных и ASCII символов (если размер слова кратен 8-ми битам). Схема расположения шестнадцатеричных чисел зависит от размеров памяти. Например, если длина слова равна 16-ти битам, редактор Нех выводит данные в столбцах из слов, которые, в свою очередь, состоят из столбцов байтов (рисунок 16-4).

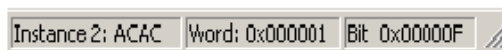
Figure 16–4. Editing 16-Bit Memory Words Using the Hex Editor



Нечитабельные символы ASCII отображаются в виде (.). Цвет данных изменяется, когда вы выполните чтение и запись. Данные, отображаемые чёрным, показывают те данные в редакторе Нех, которые остались такими же, что и данные, прочитанные из чипа. Если данные в редакторе Нех изменили цвет на красный, то предыдущие данные в редакторе Нех отличаются от данных, прочитанных из чипа.

Когда вы анализируете данные, вы можете использовать курсор и панель статуса для быстрой идентификации точной локализации байта в памяти. Строка статуса расположена внизу редактора содержимого памяти в системе, она показывает выбранное имя элемента, положение слова и начальный номер бита (рисунок 16-5).

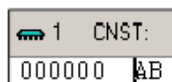
Figure 16–5. Status Bar in the In-System Memory Content Editor



Начальный номер бита – это позиция бита на курсоре внутри слова. В следующем примере, слово имеет 8-ми битную ширину.

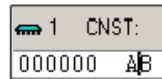
С помощью курсора в позиции, показанной на рисунке 16-6, положение слова - 0x0000, а позиция бита - 0x0007.

Figure 16–6. Hex Editor Cursor Positioned at Bit 0x0007



С помощью курсора в позиции, показанной на рисунке 16-7, положение слова осталось 0x0000, но позиция бита - 0x0003.

Figure 16–7. Hex Editor Cursor Positioned at Bit 0x0003



Поддержка скриптов

Редактор содержимого памяти в системе поддерживает чтение и запись содержимого памяти посредством Tcl скриптов или Tcl команд, вводимых из командной строки. За подробной информацией об опциях команд скриптирования, обратитесь к браузеру помощи командной строки Quartus II и Tcl API.

Для запуска браузера помощи, наберите следующую команду в командной строке:

```
quartus_sh --qhhelp ↵
```

Ссылка на руководство пользователя о скриптировании в программе Quartus II содержит ту же информацию.

За подробной информацией о Tcl скриптировании, обратитесь к главе "Tcl скриптирование" в томе 2 "Настольной книги Quartus II". За подробной информацией о скриптировании в командной строке, обратитесь к главе "Скриптирование в командной строке" в томе 2 "Настольной книги Quartus II".

Основные команды, используемые в редакторе содержимого памяти в системе, следующие:

■ Прочитать из памяти:

```
read_content_from_memory  
[-content_in_hex]  
-instance_index <instance index>  
-start_address <starting address>  
-word_count <word count>
```

■ Записать в память:

```
write_content_to_memory
```

■ Сохранить в файл содержимое памяти:

```
save_content_from_memory_to_file
```

■ Обновить содержимое памяти из файла:

```
update_content_to_memory_from_file
```

За описанием опций командной строки и примерами скриптирования, обратитесь к браузеру помощи командной строки Quartus II и Tcl API и ссылке на руководство пользователя о скриптировании в программе Quartus II.

Программирование чипа, используя редактор содержимого памяти в системе

После того, как вы сделаете изменения в вашем проекте, вы можете запрограммировать чип в редакторе содержимого памяти в системе. Для программирования чипа, выполните следующие шаги:

1. В меню **Инструменты**, кликните **Редактор содержимого памяти в системе**;
2. На панели **Цепь конфигурации JTAG** в редакторе содержимого памяти в системе, выберите файл объекта SRAM (.sof), который содержит информацию о модифицируемых элементах памяти и констант;

3. Кликните **Сканировать цепь**;
4. В списке **Чипы**, выберите чип, который вы хотите программировать;
5. Кликните **Программировать чип**.

Пример использования редактора содержимого памяти в системе вместе с встроенным логическим анализатором SignalTap II

Следующий сценарий описывает, как вы можете использовать средство обновления в системе памяти и констант совместно со встроенным логическим анализатором SignalTap II, для эффективной отладки вашего проекта в системе. Несмотря на то, что и редактор содержимого памяти в системе и встроенный логический анализатор SignalTap II используют связь по интерфейсу JTAG, вы можете применять их совместно.

После компиляции вашего FPGA проекта, вы находите, что характеристики вашего проекта *FIR filter* не такие, как ожидалось.

1. Для локализации источника проблемы, измените все ваши коэффициенты для *FIR filter* на модифицируемые в системе и установите встроенный логический анализатор SignalTap II.
2. Используя встроенный логический анализатор SignalTap II для отведения и переключения внутренних узлов проекта, вы находите, что *FIR filter* функционирует за пределами существующей верхней границы частоты.
3. Используя редактор содержимого памяти в системе, вы проверяете корректность коэффициентов *FIR filter*. При последовательном чтении коэффициентов, вы обнаруживаете, что один коэффициент некорректный.
4. Поскольку ваши коэффициенты модифицируются в системе, вы обновляете коэффициенты корректными данными, используя редактор содержимого памяти в системе.

В этом сценарии, вы добились быстрой локализации источника проблемы, используя редактор содержимого памяти в системе и встроенный логический анализатор SignalTap II. Вы также добились верификации функциональности вашего чипа, изменением значений коэффициентов, прежде чем модифицировать исходные файлы проекта.

Выводом из этого примера может быть возможность модифицировать коэффициенты в редакторе содержимого памяти в системе для варьирования характеристиками *FIR filter* (например, ослабление фильтра, перестройка полосы частот, верхней границы частоты и функции обработки методом окна).

Заключение

Средство обновления в системе памяти и констант осуществляет доступ к чипу для эффективной отладки в лаборатории. Вы можете использовать обновление в системе памяти и констант совместно с встроенным логическим анализатором SignalTap II для максимизирования эффекта проникновения внутрь чипа Altera FPGA. Наглядность метода доступа к внутренней логике чипа, позволяет вам очень просто идентифицировать и устранять проблемы в вашем проекте.