The hard (on-chip) memory interface components are available in the Arria V and Cyclone V device families. The Arria V device family includes hard memory interface components supporting DDR2 and DDR3 SDRAM memory protocols at speeds of up to 533 MHz. For the Quartus II software version 12.0 and later, the Cyclone V device family supports both hard and soft interface support.

The Arria V device family supports both hard and soft interfaces for DDR3 and DDR2, and soft interfaces for LPDDR2 SDRAM, QDR II SRAM, and RLDRAM II memory protocols. The Cyclone V device family supports both hard and soft interfaces for DDR3, DDR2, and LPDDR2 SDRAM memory protocols.

The hard memory interface consists of three main parts, as follows:

- The multi-port front end (MPFE), which allows multiple independent accesses to the hard memory controller.
- The hard memory controller, which initializes, refreshes, manages, and communicates with the external memory device.
- The hard PHY, which provides the physical layer interface to the external memory device.
Multi-Port Front End (MPFE)

The multi-port front end and its associated fabric interface provide up to six command ports, four read-data ports and four write-data ports, through which user logic can access the hard memory controller. Each port can be configured as read only or write only, or read and write ports may be combined to form bidirectional data ports. Ports can be 32, 64, 128, or 256 data bits wide, depending on the number of ports used and the type (unidirectional or bidirectional) of the port.

Fabric Interface

The fabric interface provides communication between the Avalon-ST-like internal protocol of the hard memory interface and the external Avalon-MM protocol. The fabric interface supports frequencies in the range of 10 MHz to one-half of the memory interface frequency. For example, for an interface running at 533 MHz, the maximum user logic frequency is 267 MHz. The MPFE handles the clock crossing between user logic and the hard memory interface.

The multi-port front end read and write FIFO depths are 8, and the command FIFO depth is 4. The FIFO depths are not configurable.

Operation Ordering

Requests arriving at a given port are executed in the order in which they are received.
Requests arriving at different ports have no guaranteed order of service, except when a first transaction has completed before the second arrives.

Multi-port Scheduling

Multi-port scheduling is governed by two considerations: the absolute priority of a request and the weighting of a port. User-configurable priority and weight settings determine the absolute and relative scheduling policy for each port.

Port Scheduling

The evaluation of absolute priority ensures that ports carrying higher-priority traffic are served ahead of ports carrying lower-priority traffic. The scheduler recognizes eight priority levels, with higher values representing higher priorities. Priority is absolute; for example, any transaction with priority seven will always be scheduled before transactions of priority six or lower.

When ports carry traffic of the same absolute priority, relative priority is determined based on port weighting. Port weighting is a five-bit value, and is determined by a weighted round robin (WRR) algorithm.

The scheduler can alter priority if the latency target for a transaction is exceeded. The scheduler tracks latency on a per-port basis, and counts the cycles that a transaction is pending. Each port has a priority escalation register and a pending counter engagement register. If the number of cycles in the pending counter engagement register elapse without a pending transaction being served, that transaction’s priority is escalated.

To ensure that high-priority traffic is served quickly and that long and short bursts are effectively interleaved on ports, bus transactions longer than a single DRAM burst are scheduled as a series of DRAM bursts, with each burst arbitrated separately.

The scheduler uses a form of deficit round robin (DRR) scheduling algorithm which corrects for past over-servicing or under-servicing of a port. Each port has an associated weight which is updated every cycle, with a user-configured weight added to it and the amount of traffic served subtracted from it. The port with the highest weighting is considered the most eligible.

To ensure that lower priority ports do not build up large running weights while higher priority ports monopolize bandwidth, the hard memory controller’s DRR weights are updated only when a port matches the scheduled priority. Hence, if three ports have traffic, two being priority 7 and one being priority 4, the weights for both ports at priority 7 are updated but the port with priority 4 remains unchanged.

The scheduler can be configured to lock onto a given port for a specified number of transactions when the scheduler schedules traffic at that priority level. The number of transactions is configurable on a per-port basis. For ports with large numbers of sequential addresses, you can use this feature to allow efficient open page accesses without risk of the open page being pushed out by other transactions.

DRAM Burst Scheduling

DRAM burst scheduling recognizes addresses that access the same column/row combination—also known as open page accesses. Such operations are always served in the order in which they are received in the single-port controller.

Selection of DRAM operations is a two-stage process; first, each pending transaction must wait for its timers to be eligible for execution, then the transaction arbitrates against other transactions that are also eligible for execution.

The following rules govern transaction arbitration:

- High priority operations take precedence over lower priority operations
• If multiple operations are in arbitration, read operations have precedence over write operations
• If multiple operations still exist, the oldest is served first

A high-priority transaction in the DRAM burst scheduler wins arbitration for that bank immediately if the bank is idle and the high-priority transaction’s chip select/row/column address does not match an address already in the single-port controller. If the bank is not idle, other operations to that bank yield until the high-priority operation is finished. If the address matches another chip select/row/column, the high-priority transaction yields until the earlier transaction is completed.

You can force the DRAM burst scheduler to serve transactions in the order that they are received, by setting a bit in the register set.

DRAM Power Saving Modes

The hard memory controller supports two DRAM power-saving modes: self-refresh, and fast/slow all-bank precharge powerdown exit. Engagement of a DRAM power saving mode can occur due to inactivity, or in response to a user command.

The user command to enter power-down mode forces the DRAM burst-scheduling bank-management logic to close all banks and issue the power-down command. You can program the controller to power down when the DRAM burst-scheduling queue is empty for a specified number of cycles; the DRAM is reactivated when an active DRAM command is received.

MPFE Signal Descriptions

The following table describes the signals for the multi-port front end.

Table 3-1: MPFE Signals

<table>
<thead>
<tr>
<th>Signal</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>avl_&lt;signal_name&gt;_#</td>
<td>—</td>
<td>Local interface signals.</td>
</tr>
<tr>
<td>mp_cmd_clk_#{clk}</td>
<td>Input</td>
<td>Clock for the command FIFO buffer. Follow Avalon-MM master frequency. Maximum frequency is one-half of the interface frequency, and subject to timing closure.</td>
</tr>
<tr>
<td>mp_cmd_reset_n_#{reset_n}</td>
<td>Input</td>
<td>Asynchronous reset signal for command FIFO buffer.</td>
</tr>
<tr>
<td>mp_rfifo_clk_#{clk}</td>
<td>Input</td>
<td>Clock for the read data FIFO buffer. Follow Avalon-MM master frequency. Maximum frequency is one-half of the interface frequency, and subject to timing closure.</td>
</tr>
<tr>
<td>mp_rfifo_reset_n_#{reset_n}</td>
<td>Input</td>
<td>Asynchronous reset signal for read data FIFO buffer.</td>
</tr>
<tr>
<td>mp_wfifo_clk_#{clk}</td>
<td>Input</td>
<td>Clock for the write data FIFO buffer. Follow Avalon-MM master frequency. Maximum frequency is one-half of the interface frequency, and subject to timing closure.</td>
</tr>
<tr>
<td>mp_wfifo_reset_n_#{reset_n}</td>
<td>Input</td>
<td>Asynchronous reset signal for write data FIFO buffer.</td>
</tr>
<tr>
<td>Signal</td>
<td>Direction</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>-----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>bonding_in_1/2/3</td>
<td>Input</td>
<td>Bonding interface input port. Connect second controller bonding output port to this port according to the port sequence.</td>
</tr>
<tr>
<td>bonding_out_1/2/3</td>
<td>Output</td>
<td>Bonding interface output port. Connect this port to the second controller bonding input port according to the port sequence.</td>
</tr>
</tbody>
</table>

Notes to Table:

1. # represents the number of the slave port. Values are 0—5.
2. # represents the number of the slave port. Values are 0—3.
3. The command FIFO buffers have two stages. The first stage is 4 bus transaction deep per port. After port scheduling, the commands are placed in the second stage FIFO buffer, which is 8 DRAM transactions deep. The second stage FIFO buffer is used to optimize memory operations where bank look ahead and data reordering occur. The write data buffer is 32 deep and read buffer is 64 deep.

Every input interface (command, read data, and write data) has its own clock domain. Each command port can be connected to a different clock, but the read data and write data ports associated with a command port must connect to the same clock as that command port. Each input interface uses the same reset signal as its clock.

By default, the IP generates all clock signals regardless of the MPFE settings, but all unused ports and FIFO buffers are connected to ground.

The command ports can be used only in unidirectional configurations, with either 4 write and 2 read, 3 write and 3 read, or 2 write and 4 read scenarios. For bidirectional ports, the number of clocks is reduced from 6 to a maximum of 4.

For the scenario depicted in the following figure:

- command port 0 is associated with read and write data FIFO 0 and 1
- command port 1 is associated with read data FIFO 2
- command port 2 is associated with write data FIFO 2
Therefore, if port 0 (avl_0) is clocked by a 100 MHz clock signal, mp_cmd_clk_0, mp_rfifo_clk_0, mp_rfifo_clk_1, mp_wfifo_clk_0, and mp_wfifo_clk_1 must all be connected to the same 100 MHz clock, as illustrated below.
Hard Memory Controller

The hard memory controller initializes, refreshes, manages, and communicates with the external memory device.
The hard memory controller is functionally similar to the High Performance Controller II (HPC II). For information on signals, refer to the Functional Description—HPC II Controller chapter.

Related Information

Functional Description - HPC II Controller

Clocking

The ports on the MPFE can be clocked at different frequencies, and synchronization is maintained by cross-domain clocking logic in the MPFE.

Command ports can connect to different clocks, but the data ports associated with a given command port must be attached to the same clock as that command port. For example, a bidirectional command port that performs a 64-bit read/write function has its read port and write port connected to the same clock as the command port. Note that these clocks are separate from the EMIF core generated clocks.

Reset

The ports of the MPFE must connect to the same reset signal.

When the reset signal is asserted, it resets the command and data FIFO buffer in the MPFE without resetting the hard memory controller.

Note: The `global_reset_n` and `soft_reset_n` signals are asynchronous.

At power-up, follow the following sequence for reset signals:

1. Initially `global_reset_n`, `soft_reset_n`, and the MPFE reset signals are all asserted.
2. `global_reset_n` is deasserted.
3. Wait for `pll_locked` to transition high.
4. `soft_reset_n` is deasserted.
5. Wait for the controller signal `local_cal_success` to go high, indicating that the external memory interface has successfully completed calibration.
6. Deassert the MPFE FIFO reset signals, to ensure that no read/write activity occurs without successful interface calibration.

DRAM Interface

The DRAM interface is 40 bits wide, and can accommodate 8-bit, 16-bit, 16-bit plus ECC, 32-bit, or 32-bit plus ECC configurations. Any unused I/Os in the DRAM interface can be reused as user I/Os.

The DRAM interface supports DDR2 and DDR3 memory protocols, and LPDDR2 for Cyclone V only. Fast and medium speed grade devices are supported to 533 MHz for Arria V and 400 MHz for Cyclone V.

ECC

The hard controller supports both error-correcting code (ECC) calculated by the controller and by the user. Controller ECC code employs standard Hamming logic which can detect and correct single-bit errors and detect double-bit errors. The controller ECC is available for 16-bit and 32-bit widths, each requiring an additional 8 bits of memory, resulting in an actual memory width of 24-bits and 40-bits, respectively.

In user ECC mode, all bits are treated as data bits, and are written to and read from memory. User ECC can implement nonstandard memory widths such as 24-bit or 40-bit, where ECC is not required.
Controller ECC

Controller ECC provides the following features:

**Byte Writes**—The memory controller performs a read/modify/write operation to keep ECC valid when a subset of the bits of a word is being written. If an entire word is being written (but less than a full burst) and the DM pins are connected, no read is necessary and only that word is updated. If controller ECC is disabled, byte-writes have no performance impact.

**ECC Write Backs**—When a read operation detects a correctable error, the memory location is scheduled for a read/modify/write operation to correct the single-bit error.

**User ECC**—User ECC is 24-bits or 40-bits wide; with user ECC, the controller performs no ECC checking. The controller employs memory word addressing with byte enables, and can handle arbitrary memory widths. User ECC does not disable byte writes; hence, you must ensure that any byte writes do not result in corrupted ECC.

Bonding of Memory Controllers

Bonding is a feature that allows data to be split between two memory controllers, providing the ability to service bandwidth streams similar to a single 64-bit controller. Bonding works by dividing data buses in proportion to the memory widths, and always sending a transaction to both controllers. When signals are returned, bonding ensures that both sets of signals are returned identically.

Bonding can be applied to asymmetric controllers, and allows controllers to have different memory clocks. Bonding does not attempt to synchronize the controllers. Bonding supports only one port. The Avalon port width can be varied from 64-bit to 256-bit; 32-bit port width is not supported.

The following signals require bonding circuitry:

**Read data return**—This bonding allows read data from the two controllers to return with effectively one ready signal to the bus master that initiated the bus transaction.

**Write ready**—For Avalon-MM, this is effectively bonding on the waitrequest signal.

**Write acknowledge**—Synchronization on returning the write completed signal.

For each of the above implementations, data is returned in order, hence the circuitry must match up for each valid cycle.

Bonded FIFO buffers must have identical FIFO numbers; that is, read FIFO 1 on controller 1 must be paired with Read FIFO 1 on controller 2.

Data Return Bonding

Long loop times can lead to communications problems when using bonded controllers. The following effects are possible when using bonded controllers:

- If one memory controller completes its transaction and receives new data before the other controller, then the second controller can send data as soon as it arrives, and before the first controller acknowledges that the second controller has data.
- If the first controller has a single word in its FIFO buffer and the second controller receives single-word transactions, the second controller must determine whether the second word is a valid signal or not.

To accommodate the above effects, the hard controller maintains two counters for each bonded pair of FIFO buffers and implements logic that monitors those counters to ensure that the bonded controllers receive the same data on the same cycle, and that they send the data out on the same cycle.
FIFO Ready

FIFO ready bonding is used for write command and write data buses. The implementation is similar to the data return bonding.

Bonding Latency Impact

Bonding has no latency impact on ports that are not bonded.

Bonding Controller Usage

Arria V and Cyclone V devices employ three shared bonding controllers to manage the read data return bonding, write acknowledge bonding, and command/write data ready bonding.

The three bonding controllers require three pairs of bonding I/Os, each based on a six port count; this means that a bonded hard memory controller requires 21 input signals and 21 output signals for its connection to the fabric, and another 21 input signals and 21 output signals to the paired hard memory controller.

Note: The hard processor system (HPS) hard memory controller cannot be bonded with another hard memory controller on the FPGA portion of the device.

Bonding Configurations and Parameter Requirements

Altera has verified hard memory controller bonding between two interfaces with the following configuration:

- Same clock source
- Same memory clock frequency
- Same memory parameters and timings (except interface width)
- Same controller settings.
- Same port width in MPFE settings

Bonding supports only one port. The Avalon port width can be varied from 64-bits to 256-bits; a 32-bit port width is not supported.

Hard PHY

A physical layer interface (PHY) is embedded in the periphery of the Arria V device, and can run at the same high speed as the hard controller and hard sequencer. This hard PHY is located next to the hard controller. Differing device configurations have different numbers and sizes of hard controller and hard PHY pairs.

The hard PHY implements logic that connects the hard controller to the I/O ports. Because the hard controller and AFI interface support high frequencies, a portion of the sequencer is implemented as hard logic. The Nios II processor, the instruction/data RAM, and the Avalon fabric of the sequencer are implemented as core soft logic. The read/write manger and PHY manager components of the sequencer, which must operate at full rate, are implemented as hard logic in the hard PHY.

Interconnections

The hard PHY resides on the device between the hard controller and the I/O register blocks. The hard PHY is instantiated or bypassed entirely, depending on the parameterization that you specify.

The hard PHY connects to the hard memory controller and the core, enabling the use of either the hard memory controller or a software-based controller. (You can have the hard controller and hard PHY, or the soft controller and soft PHY; however, the combination of soft controller with hard PHY is not supported.) The hard PHY also connects to the I/O register blocks and the DQS logic. The path between the hard PHY
and the I/O register blocks can be bypassed, but not reconfigured—in other words, if you use the hard PHY datapath, the pins to which it connects are predefined and specified by the device pin table.

**Clock Domains**

The hard PHY contains circuitry that uses the following clock domains:

- **AFI clock domain (pll_afi_clk)** — The main full-rate clock signal that synchronizes most of the circuit logic.

- **Avalon clock domain (pll_avl_clk)** — Synchronizes data on the internal Avalon bus, namely the Read/Write Manager, PHY Manager, and Data Manager data. The data is then transferred to the AFI clock domain. To ensure reliable data transfer between clock domains, the Avalon clock period must be an integer multiple of the AFI clock period, and the phases of the two clocks must be aligned.

- **Address and Command clock domain (pll_addr_cmd_clk)** — Synchronizes the global asynchronous reset signal, used by the I/Os in this clock domain.

**Hard Sequencer**

The sequencer initializes the memory device and calibrates the I/Os, with the objective of maximizing timing margins and achieving the highest possible performance.

When the hard memory controller is in use, a portion of the sequencer must run at full rate; for this reason, the Read/Write Manager, PHY Manager, and Data Manager are implemented as hard components within the hard PHY. The hard sequencer communicates with the soft-logic sequencer components (including the Nios II processor) via an Avalon bus.

**MPFE Setup Guidelines**

The following instructions provide information on configuring the multi-port front end of the hard memory interface.

1. To enable the hard memory interface, turn on **Enable Hard External Memory Interface** in the **Interface Type** tab in the parameter editor.

2. To export bonding interface ports to the top level, turn on **Export bonding interface** in the **Multiple Port Front End** pulldown on the **Controller Settings** tab in the parameter editor.

   **Note:** The system exports three bonding-in ports and three bonding-out ports. You must generate two controllers and connect the bonding ports manually.

3. To expand the interface data width from a maximum of 32 bits to a maximum of 40 bits, turn on **Enable Avalon-MM data for ECC** in the **Multiple Port Front End** pulldown on the **Controller Settings** tab in the parameter editor.

   **Note:** The controller does not perform ECC checking when this option is turned on.

4. Select the required **Number of ports** for the multi-port front end in the **Multiple Port Front End** pulldown on the **Controller Settings** tab in the parameter editor.

   **Note:** The maximum number of ports is 6, depending on the port type and width. The maximum port width is 256 bits, which is the maximum data width of the read data FIFO and write data FIFO buffers.

5. The table in the **Multiple Port Front End** pulldown on the **Controller Settings** tab in the parameter editor lists the ports that are created. The columns in the table describe each port, as follows:

   - **Port**: Indicates the port number.
• **Type**: Indicates whether the port is read only, write only, or bidirectional.

• **Width**: To achieve optimum MPFE throughput, Altera recommends setting the MPFE data port width according to the following calculation:

\[
2 \times (\text{frequency ratio of HMC to user logic}) \times (\text{interface data width})
\]

For example, if the frequency of your user logic is one-half the frequency of the hard memory controller, you should set the port width to be 4x the interface data width. If the frequency ratio of the hard memory controller to user logic is a fractional value, you should use a larger value; for example, if the ratio is 1.5, you can use 2.

• **Priority**: The priority setting specifies the priority of the slave port, with higher values representing higher priority. The slave port with highest priority is served first.

• **Weight**: The weight setting has a range of values of 0–31, and specifies the relative priority of a slave port, with higher weight representing higher priority. The weight value can determine relative bandwidth allocations for slave ports with the same priority values. For example, if two ports have the same priority value, and weight values of 4 and 6, respectively, the port with a weight of 4 will receive 40% of the bus bandwidth, while the port with a weight of 6 will receive 60% of the bus bandwidth—assuming 100% total available bus bandwidth.

### Soft Memory Interface to Hard Memory Interface Migration Guidelines

The following instructions provide information on mapping your soft memory interface to a hard memory interface.

#### Pin Connections

1. The hard and soft memory controllers have compatible pinouts. Assign interface pins to the hard memory interface according to the pin table.

2. Ensure that your soft memory interface pins can fit into the hard memory interface. The hard memory interface can support a maximum of a 40-bit interface with user ECC, or a maximum of 80-bits with same-side bonding. The soft memory interface does not support bonding.

3. Follow the recommended board layout guidelines for the hard memory interface.

#### Software Interface Preparation

Observe the following points in preparing your soft memory interface for migration to a hard memory interface:

• You cannot use the hard PHY without also using the hard memory controller.

• The hard memory interface supports only full-rate controller mode.

• Ensure that the MPFE data port width is set according to the soft memory interface half-rate mode Avalon data width.

• The hard memory interface uses a different Avalon port signal naming convention than the software memory interface. Ensure that you change the `avl_<signal_name>` signals in the soft memory interface to `.avl_<signal_name>_0` signals in the hard memory interface.

• The hard memory controller MPFE includes an additional three clocks and three reset ports (CMD port, RFIFO port, and WFIFO port) that do not exist with the soft memory controller. You should connect the user logic clock signal to the MPFE clock port, and the user logic reset signal to the MPFE reset port.

• In the soft memory interface, the half-rate `afi_clk` is a user logic clock. In the hard memory interface, `afi_clk` is a full-rate clock, because the core fabric might not be able to achieve full-rate speed. When you migrate your soft memory interface to a hard memory interface, you need to supply an additional slower rate clock. The maximum clock rate supported by core logic is one-half of the maximum interface frequency.
Latency

Overall, you should expect to see slightly more latency when using the hard memory controller and multi-port front end, than when using the soft memory controller.

The hard memory controller typically exhibits lower latency than the soft memory controller; however, the multi-port front end does introduce additional latency cycles due to FIFO buffer stages used for synchronization. The MPFE cannot be bypassed, even if only one port is needed.

Bonding Interface Guidelines

Bonding allows a single data stream to be split between two memory controllers, providing the ability to expand the interface data width similar to a single 64-bit controller. This section provides some guidelines for setting up the bonding interface.

1. Bonding interface ports are exported to the top level in your design. You should connect each bonding_in* port in one hard memory controller to the corresponding bonding_out_* port in the other hard memory controller, and vice versa.

2. You should modify the Avalon signal connections to drive the bonding interface with a single user logic/master, as follows:
   a. AND both avl_ready signals from both hard memory controllers before the signals enter the user logic.
   b. AND both avl_rdata_valid signals from both hard memory controllers before the signals enter the user logic. (The avl_rdata_valid signals should be identical for both hard memory controllers.)
   c. Branch the following signals from the user logic to both hard memory controllers:
      - avl_burstbegin
      - avl_addr
      - avl_read_req
      - avl_write_req
      - avl_size
   d. Split the following signals according to each multi-port front end data port width:
      - avl_rdata
      - avl_wdata
      - avl_be

Document Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>August 2014</td>
<td>2014.08.15</td>
<td>• Updated descriptions of  mp_cmd_reset_n_#<em>reset_n, mp_rfifo_reset_n</em>#<em>reset_n, and mp_wfifo_reset_n</em>#_reset_n in the MPFE Signals table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added Reset section to Hard Memory Controller section.</td>
</tr>
<tr>
<td>Date</td>
<td>Version</td>
<td>Changes</td>
</tr>
<tr>
<td>---------------</td>
<td>---------</td>
<td>-------------------------------------------------------------------------</td>
</tr>
<tr>
<td>December 2013</td>
<td>2013.12.16</td>
<td>- Added footnote about command FIFOs to MPFE Signals table.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Added information about FIFO depth for the MPFE.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Added information about hard memory controller bonding.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Reworded protocol-support information for Arria V and Cyclone V devices.</td>
</tr>
<tr>
<td>November 2012</td>
<td>2.1</td>
<td>- Added <em>Hard Memory Interface Implementation Guidelines</em>.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Moved content of EMI-Related HPS Features in SoC Devices section to chapter 4, <em>Functional Description—HPS Memory Controller</em>.</td>
</tr>
<tr>
<td>June 2012</td>
<td>2.0</td>
<td>- Added EMI-Related HPS Features in SoC Devices.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Added LPDDR2 support.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Added Feedback icon.</td>
</tr>
<tr>
<td>November 2011</td>
<td>1.0</td>
<td>Initial release.</td>
</tr>
</tbody>
</table>