Sharing External Memory Bandwidth Using the Multi-Port Front-End Reference Design

AN-637-01 Application Note

This document describes the features and architecture of the Altera® Multi-Port Front-End (MPFE) reference design, details the design flow you should follow to integrate the MPFE block into your design, and illustrates the functionality of the MPFE block in an example system with multiple masters.

The MPFE reference design allows you to efficiently share access to external memory between multiple data masters in your design. Combined with the Altera DDR2 and DDR3 SDRAM Controller with UniPHY, the MPFE block allows efficient access to DDR2 or DDR3 SDRAM memories.

In this document, the terms master and slave are relative to the MPFE reference design, unless specifically referred to as a master in your system. The master port on the MPFE block connects to the slave port on the memory controller. Each slave port on the MPFE connects to the master port on the user logic block in your system that requires access to external memory.

Multi-Port Front-End Block

Multiple functional blocks in a system can share a single external memory interface. Sharing access to the same external memory interface requires an arbiter that manages memory read and write requests from multiple functional blocks. You can use Qsys or SOPC Builder system integration tools to create arbitration logic that uses a weighted round-robin scheme. However, the arbitration logic generated by these tools does not support prioritization of requests from one data master over another.

The MPFE reference design features a priority weighted round-robin arbitration scheme that allows you to control the traffic flow to and from the external memory interface. By defining which ports are critical, you can ensure that the time-critical masters in your system, such as video and audio blocks, have priority over other less time sensitive blocks. It also allows the non-critical masters to use any available bandwidth in times when the critical masters are not requesting access. Systems that contain both high and low priority masters can use the MPFE component to efficiently share memory bandwidth. While the MPFE block is optimized for high data rate applications such as video processing designs, it also supports small, random address accesses such as from a processor.

The MPFE reference design has the following features:

- Multi-class, weighted round-robin arbiter
  - Allows masters to be classified as critical and non-critical to protect your time-critical processing blocks
  - Allows you to assign bandwidth weights to distribute the memory bandwidth between the masters in your system

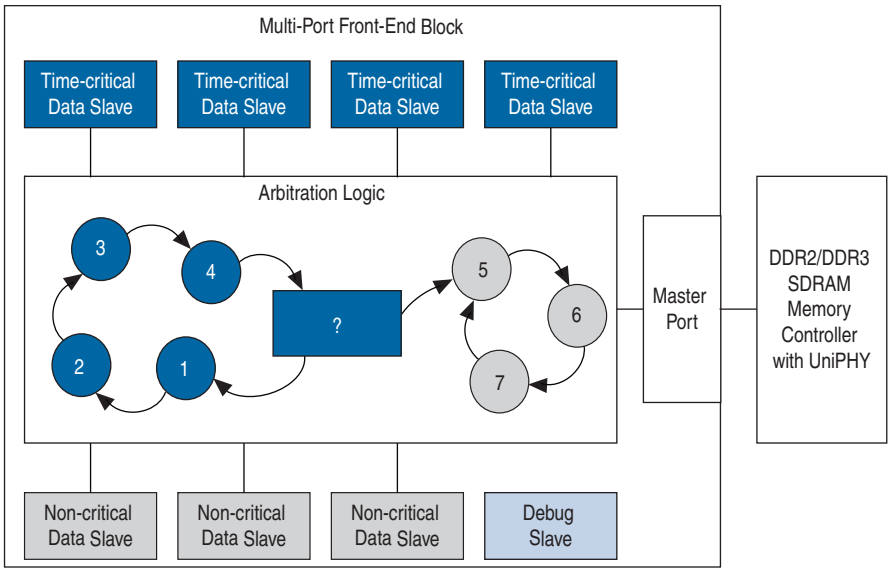
101 Innovation Drive
San Jose, CA 95134
www.altera.com

- Support for up to 16 Avalon® Memory-Mapped (Avalon-MM) read and write ports
  - Full width ports that can use the full memory bandwidth when granted
  - Width-adapting ports that allow you to connect 32-bit masters to the wider external memory interface efficiently
- Support for a wide range of configurations
  - Up to 512-bit Avalon-MM interfaces
  - All JEDEC memory configurations up to 4 GB of external memory
  - Efficient bursts of up to 64 beats
- Real-time in-system performance monitoring through a debug slave
- Support for memory controllers that feature an Avalon-MM port
  - High-Performance Controller II with UniPHY and ALTMEMPHY
- Support for the following Altera devices:
  - Arria® II GX, Arria II GZ, Cyclone® III, Cyclone IV, Stratix® III, and Stratix IV
- Optimized operation of 267 MHz in Stratix III and Stratix IV devices
- Provided as clear text register transfer level (RTL)
- Easy-to-use GUI

Functional Description

Figure 1 shows a block diagram of the MPFE reference design. The block diagram shows the three different types of slave ports that are available: time-critical data slaves, non-critical data slaves, and a debug slave. You can parameterize each of the 16 available data slave ports to be a time-critical or non-critical port.

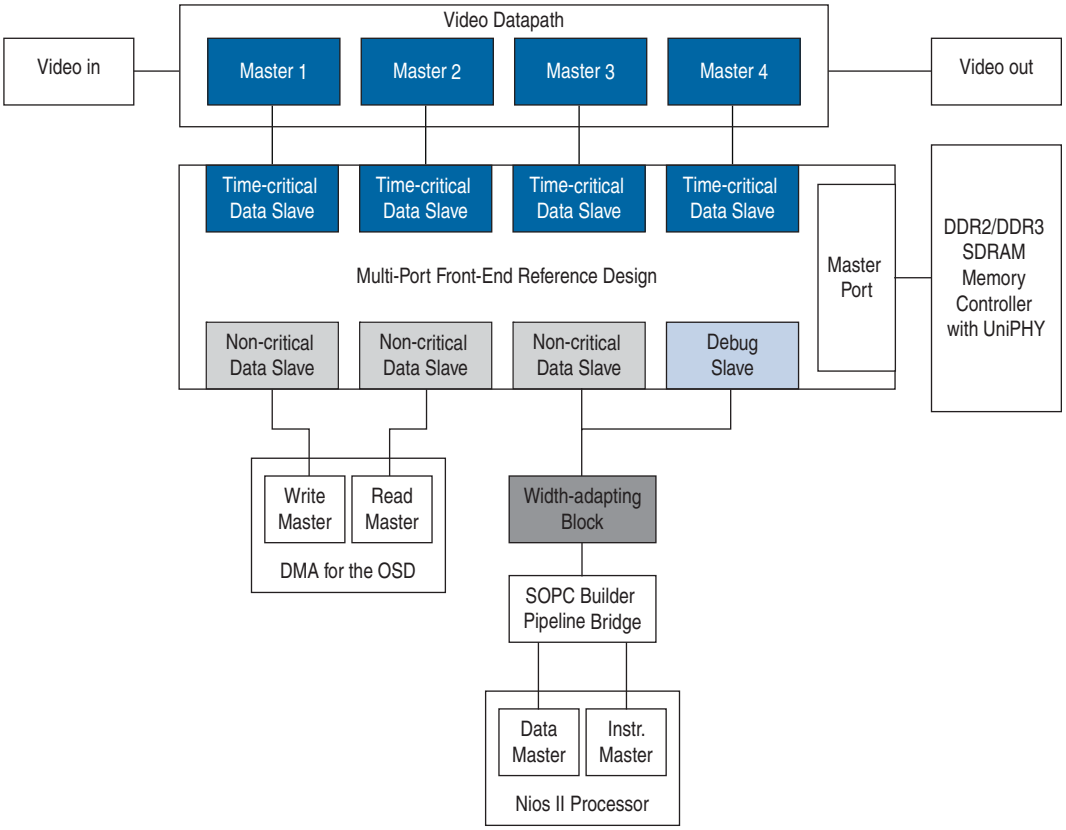
Figure 1. Block Diagram of the MPFE Reference Design



Each data slave port connects to a master in your system. You can configure the data width of 14 of the 16 ports to match the memory interface local data width. The remaining 2 ports are fixed 32-bit wide width-adapting ports. You can connect these two ports to 32-bit masters such as a Nios® II processor, and they support a burst adaptor that automatically merges multiple 32-bit bursts to a single larger burst on the memory controller interface.

Figure 2 shows an example of how you can connect the MPFE block to a video processing datapath, a direct memory access (DMA) for the on-screen display (OSD), and to a Nios II processor.

Figure 2. MPFE Block In a Video Processing System



Interface Ports

The MPFE reference design has the following Avalon-MM ports:

- Master port that connects to the memory controller.
- Data slave ports that connect to the masters in the user logic.
- Debug slave port that allows access to internal counters used for tracking performance.

Master Port

The master port supports a maximum burst size of 64, which matches the maximum burst size supported by the Avalon-MM slave port of the DDR2 and DDR3 SDRAM controllers. You should always configure the MPFE reference design to have the same maximum burst size as your memory controller to prevent Qsys and SOPC Builder from inserting any burst adaptive logic between them.

Data Slave Ports

The data slave ports, with the exception of the width-adapting ports, have the same width as the master port that connects to the memory controller. The slave ports support data widths of 32, 64, 128, 256 and 512 bits. The slave ports do not accept posted writes. The reference design supports up to a maximum of 16 data slave ports.

If your design requires the memory to be shared with more than 16 masters, you can insert a pipeline bridge in the Qsys and SOPC Builder and use the bridge to connect multiple masters to one data slave port on the MPFE reference design.

Width-Adapting Block

Data slave ports 6 and 7 have a fixed data width of 32 bits to support fixed-width masters such as the Nios II data and instruction masters. These width-adapting data ports are a power-of-two times wider than the fixed-width master port. For example, a 256-bit wide data port is 8 times wider than the 32-bit fixed-width port.

In a system where the shared slave is 32*n bits wide, each width-adapting port queues up n sequential read requests before presenting a single 32*n-bit wide read request to the arbiter. The read request then only requires a single transaction on the master port to the memory controller interface, instead of n separate requests. When the memory controller slave returns this read data, your system's master receives the read data over n clock cycles, saving n–1 cycles of the shared memory controller slave's bandwidth.

In a system with a 256-bit wide memory controller interface, the width-adapting slave queues 8 read requests from the 32-bit fixed-width port and issues a single read request to the memory controller. Without this aggregation, the MPFE component would have to issue 8 separate requests to the memory controller and discard 88% of the returned data.

For writes, a width-adapting port accepts n sequential write requests and allows the user data master to post up to n beats of write data into the port. The width-adapting port then issues the write request to the arbiter and only requires a single transaction on the master port to the memory controller interface.

The width-adapting port has a time-out mechanism to prevent the master from being locked out if it presents less than n read or write requests. The mechanism adds to the best-case latency for these accesses, but on average, the bandwidth savings to the system as a whole offsets the effects of these accesses.

Debug Port

The debug port monitors the performance of the MPFE component and provides information that you can use to optimize your system. This port features an Avalon-MM register slave interface that provides access to transaction counters and latency timers that you can use to assess and tune the performance of your system. You must read the counters and timers in the debug port once every second to prevent them from saturating. You can clear these registers by writing to address 0×0 of the debug slave.

Each slave port has counters that record the following:

- Number of times the port gets access.
- Number of words of read or write data the port receives or sends.
- Worst case latency seen on an access since the last time the counter clears.
- Number of cycles of wait state for the port.

The master port has counters that record the following:

- Number of words of read or write data the port receives or sends.
- Number of cycles of wait state from the slave (memory controller).

Clock Crossing

All the ports in this design are synchronous to the shared memory controller. The slave ports do not have clock crossing logic. Clock crossing logic increases the complexity and the latency of the MPFE component. If your design requires this functionality, implement clock crossing FIFO buffers in your master components or use the Qsys or SOPC Builder tool to automatically insert these blocks.

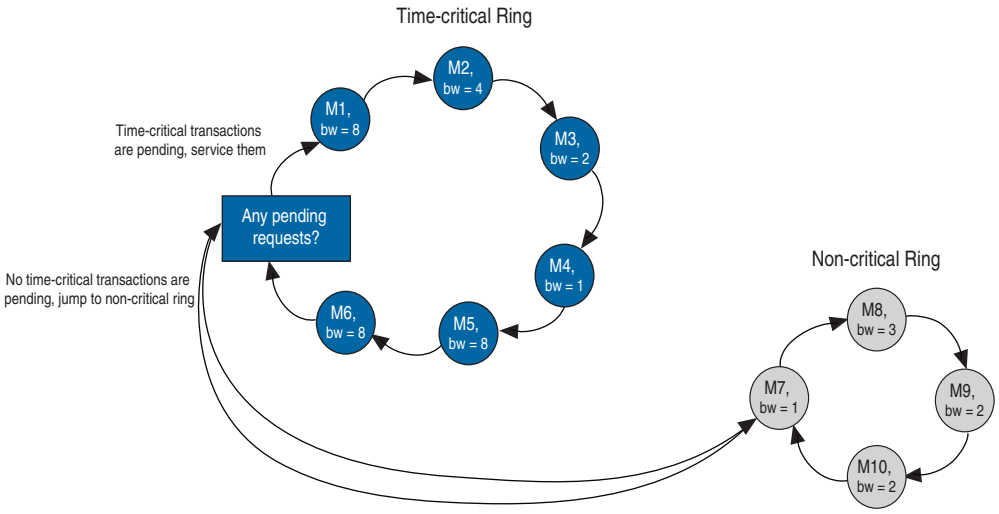
Arbitration

The MPFE reference design shares access to external memory between the different accesses presented on its slave ports. The arbiter in the MPFE reference design performs the following functions:

- Divides traffic into two classes—critical and non-critical—to protect the time-critical accesses
- Shares the available bandwidth between the slave ports using user-specified bandwidth ratios

Figure 3 shows the arbitration scheme in a system with 10 masters and their corresponding bandwidth allocation settings.

Figure 3. Arbitration Scheme



In this arbitration scheme, each data master on the time-critical ring has a share of the available bandwidth based on the settings you specify. The arbiter cycles around the data ports, granting each the ability to issue a read or write burst of up to the largest supported burst size (maximum 64 beats) to the memory controller. The arbiter continues to go around this ring, servicing slave ports that have not exceeded their bandwidth allowance.

Whenever there are no outstanding time-critical requests, the arbiter accepts the next pending transaction from the non-critical ring. Once the non-critical transaction is serviced, the arbiter checks to see if there are any new requests on the time-critical ring and returns to service that request. If more than one request is present on the time-critical ring, the arbiter continues to service them. After the arbiter services all the pending time-critical requests, it switches back to servicing the non-critical ring.

Sharing Bandwidth

The MPFE arbiter uses an enhanced version of the weighted round-robin scheme which grants access to successive slave ports in medium sized blocks (up to 64 beats), but has a leaky bucket bandwidth allocation system to distribute the accesses more evenly. Granting access to medium sized blocks, ideally less than or equal to the size of the row or page in the SDRAM memory, reduces the amount of bank management that the memory controller has to do, and distributes accesses across slave ports reduces the worst case latency. Reduced latency means you require less buffering in your design.

The ordering of grants in this system is more difficult to predict than a basic round-robin system. However, the MPFE component supports a debug register slave that enables you to observe the arbitration behavior during the operation of the system.

In the system illustrated in Figure 3, masters M1, M5, and M6 each get 8 shares of the total available bandwidth. The total number of bandwidth shares on the time-critical ring adds up to 31, and hence the M1 master is allocated 25% (8/31) of memory bandwidth. Master M2 gets 4 shares of the available bandwidth, or half of the bandwidth allocated to master M1. The masters M7 through M10 get access to the external memory when the first 6 masters are idle and not requesting memory access.

Performance

This section discusses the performance of the MPFE reference design in terms of the memory efficiency, latency, frequency of operation, and resource utilization.

Memory Efficiency

The memory interface efficiency you can achieve using the MPFE reference design depends on your traffic patterns. Use functional simulations or the information from the MPFE debug slave port to calculate the efficiency for your system.

For example, in the High Definition Video Reference Design (UDX3) attains an efficiency of more than 90% as measured using the debug port. The High Definition Video Reference Design uses the MPFE block to share access to an external memory with over 14 data masters.

For more information about the UDX3 reference design, refer to *AN 604: High Definition Video Reference Design (UDX3)*.

Latency

Because the MPFE component adds one extra cycle of latency to any request that is serviced immediately, the minimum command latency through the component is 1 clock cycle. For read transactions, the MPFE component adds an extra three cycles of latency on the data return path; data that is transferred to the slave port through a FIFO buffer. However, this latency does not include the read latency of the memory controller itself.

The worst case latency depends on the number of slave ports and the size of bursts they request. There is one cycle of delay when the slave ports switch. You can measure the latency for a specific request in functional simulation by observing the delay between the time the slave read or write request signal asserts to the time the slave waitrequest signal deasserts and or the read data valid signal asserts.

Frequency of Operation

The MPFE reference design operates up to 267 MHz, which matches the maximum core clock frequency of the half-rate 533-MHz DDR3 SDRAM Controller with UniPHY. The timing closure for the MPFE reference design was verified using the Quartus® II design software version 10.1 and targeting a Stratix IV C2 speed grade device (EP4SGX230KF40C2).

Resource Utilization

The MPFE reference design uses approximately 7,500 ALUTs, 5,000 registers, and 34 Kbits of memory when implementing all 16 slave ports with 256-bit wide data buses and the debug slave port. With the optional debug port disabled, the MFPE resource utilization drops down to 4,200 ALUTs and 2,500 registers.

Figure 4 shows a detailed breakdown of the resource utilization for the MPFE reference design.

Figure 4. MPFE Reference Design Resource Utilization

	Compilation Hierarchy Node	LC Combinationals	LC Registers	Block Memory Bits
1	⊟ |mpfe_top	7528 (84)	4933 (298)	35200
2	|mpfe_arbiter:arbiter|	1669 (1669)	525 (525)	0
3	|mpfe_debug_port:debug_port|	3307 (3307)	2509 (2509)	0
4	⊟ |mpfe_fifo:addr_fifo|	30 (0)	23 (0)	2432
5	⊟ |scfifo:scfifo_component|	30 (0)	23 (0)	2432
6	⊟ |scfifo_la61:auto_generated|	30 (0)	23 (0)	2432
7	⊟ |a_dpfifo_tn31:dpfifo|	30 (0)	23 (0)	2432
8	⊟ |a_fefifo_t7f:fifo_state|	16 (9)	9 (2)	0
9	|cntr_bl7:count_usedw|	7 (7)	7 (7)	0
10	|cntr_vkb:rd_ptr_count|	7 (7)	7 (7)	0
11	|cntr_vkb:wr_ptr|	7 (7)	7 (7)	0
12	⊟ |dpram_f211:FIFOram|	0 (0)	0 (0)	2432
13	|altsyncram_t0k1:altsyncram1|	0 (0)	0 (0)	2432
14	⊟ |mpfe_fifo:data_fifo|	31 (0)	24 (0)	32768
15	⊟ |scfifo:scfifo_component|	31 (0)	24 (0)	32768
16	⊟ |scfifo_8c61:auto_generated|	31 (2)	24 (1)	32768
17	⊟ |a_dpfifo_qp31:dpfifo|	29 (1)	23 (0)	32768
18	⊟ |a_fefifo_t7f:fifo_state|	14 (7)	9 (2)	0
19	|cntr_bl7:count_usedw|	7 (7)	7 (7)	0
20	|cntr_vkb:rd_ptr_count|	7 (7)	7 (7)	0
21	|cntr_vkb:wr_ptr|	7 (7)	7 (7)	0
22	⊟ |dpram_2411:FIFOram|	0 (0)	0 (0)	32768
23	|altsyncram_77k1:altsyncram1|	0 (0)	0 (0)	32768
24	|mpfe_master_port:master|	1621 (1621)	316 (316)	0
25	|mpfe_slave_port:slave_port_0|	1 (1)	1 (1)	0
26	|mpfe_slave_port:slave_port_10|	1 (1)	1 (1)	0
27	|mpfe_slave_port:slave_port_11|	1 (1)	1 (1)	0
28	|mpfe_slave_port:slave_port_12|	1 (1)	1 (1)	0
29	|mpfe_slave_port:slave_port_13|	1 (1)	1 (1)	0
30	|mpfe_slave_port:slave_port_14|	1 (1)	1 (1)	0
31	|mpfe_slave_port:slave_port_15|	1 (1)	1 (1)	0
32	|mpfe_slave_port:slave_port_1|	1 (1)	1 (1)	0
33	|mpfe_slave_port:slave_port_2|	1 (1)	1 (1)	0
34	|mpfe_slave_port:slave_port_3|	1 (1)	1 (1)	0
35	|mpfe_slave_port:slave_port_4|	1 (1)	1 (1)	0
36	|mpfe_slave_port:slave_port_5|	1 (1)	1 (1)	0
37	|mpfe_slave_port:slave_port_8|	1 (1)	1 (1)	0
38	|mpfe_slave_port:slave_port_9|	1 (1)	1 (1)	0
39	|mpfe_width_adapting_slave_port:slave_port_6|	386 (386)	612 (612)	0
40	|mpfe_width_adapting_slave_port:slave_port_7|	386 (386)	612 (612)	0

Design Flow

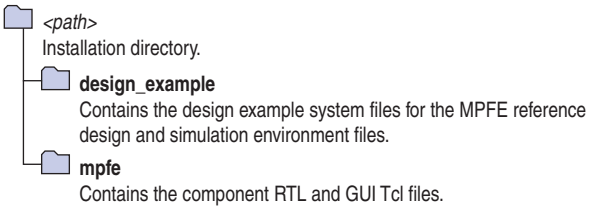
This section describes the design flow to implement the MPFE reference design in your system.

Getting Started

You can obtain the MPFE reference design from the **MPFE_Reference_Design.zip** file provided with this document. The **MPFE_Reference_Design.zip** file contains all the designs files, and a **README.pdf** file that provides information for the contents of the **mpfe** and **design_example** folders.

Unzip the **MPFE_Reference_Design.zip** file in the working directory you designate for this project. Figure 5 shows the directory structure and contents.

Figure 5. Directory Structure



The **design_example** folder contains an example system that uses the MPFE reference design to share access to a UniPHY-based DDR3 SDRAM controller with multiple instances of the Traffic Generator and Built-in Self Test (BIST) Engine module. This folder also contains the simulation environment you can use to understand and verify functionality of the MPFE reference design.

Instantiating the MPFE Reference Design

You can implement the MPFE reference design using the Qsys or SOPC Builder system integration tool, or as a standalone component in your RTL design.

To instantiate the MPFE reference design, you need to add the MPFE RTL files to your project's Qsys or SOPC Builder library. To add the MPFE reference design, create a folder named **ip** in your project directory and copy the **mpfe** folder to this new *<project_dir>***/ip** directory. The MPFE reference design will be visible in the **Component Library** under the **Project>Memories and Memory Controllers** category when you restart Qsys or SOPC Builder.

Alternatively, you can add the the MPFE reference design to your Quartus II installation directory for use in multiple projects, or add the MPFE reference design to the Qsys **IP Search Path** under the **Tools>Options** menu. With these methods, the MPFE reference design will be visible in the **Component Library** under the **Library>Memories and Memory Controllers** category as **Multi-Port Front-End** when you restart Qsys or SOPC Builder.

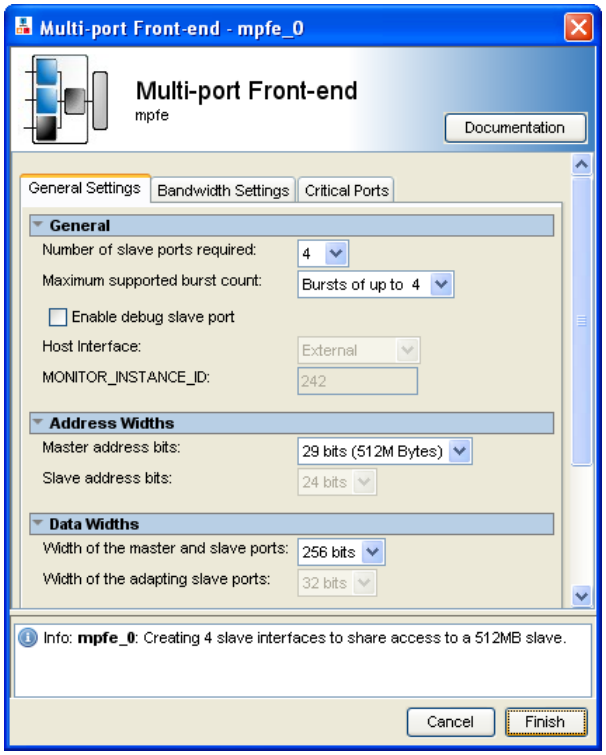
MFPE Reference Design Parameter Settings

The MPFE parameter editor has the following tabs:

- General Settings
- Bandwidth Settings
- Critical Ports

Figure 6 shows the **General Settings** tab in the MPFE parameter editor.

Figure 6. General Settings Tab



General Settings

The **General Settings** tab allows you to configure the number of ports available, the width of the ports, and the number of address bits for the ports. You can use the MPFE parameter editor to enable the debug slave port.

Table 1 lists the parameters for the **General Settings** tab.

Table 1. General Settings Parameters

Parameter	Description
Number of slave ports required	Choose the number of slave ports to enable. If you require any width adapting ports, you must choose at least 7 because only ports 6 and 7 are width-adapting ports. The width of all the other ports are configurable to match the memory controller data width.
Maximum supported burst count	Choose the maximum burst count that the MPFE reference design supports. For best results, set the maximum burst count of the arbiter and the SDRAM memory controller to the same value as your masters.
Enable debug slave port	Add a debug slave port that allows you to read data about the performance of the MPFE. Refer the "Debug Slave Registers" on page 16 for the address map of the debug slave.

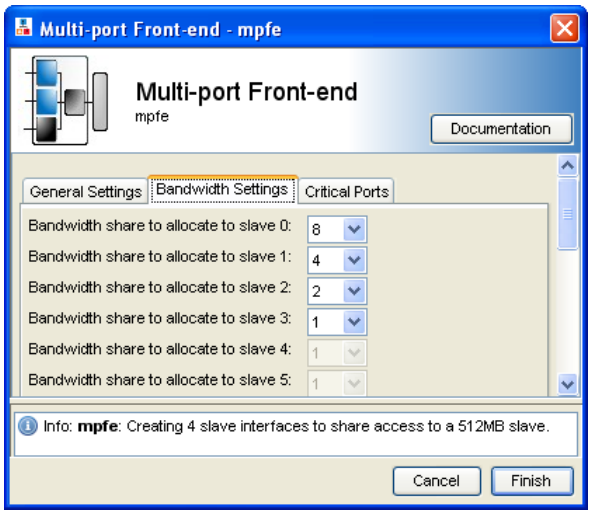
Table 1. General Settings Parameters

Parameter	Description
Master address bits	Specify the width of the address bus on the MPFE master port. This must be the same number of bytes as the memory interface that you are sharing. Note that Avalon-MM master addresses are expressed in bytes (not words), and your masters must also generate byte addresses.
Slave address bits	Specify the width of the address bus on the MPFE slave port. The width must be the same width you specify for the master address width, expressed in words instead of bytes. The parameter editor calculates this value.
Width of the master and slave ports	Choose the width of configurable data ports on the MPFE. For best results, the width of the ports on the MPFE component, the masters in your design and the DDR2 or DDR3 SDRAM memory controller slave port should all have the same width.
Width of the adapting slave ports	The width of the adapting slave port is fixed to 32 bits.

Bandwidth Settings

Figure 7 shows the **Bandwidth Settings** tab in the MPFE parameter editor.

Figure 7. Bandwidth Settings Tab



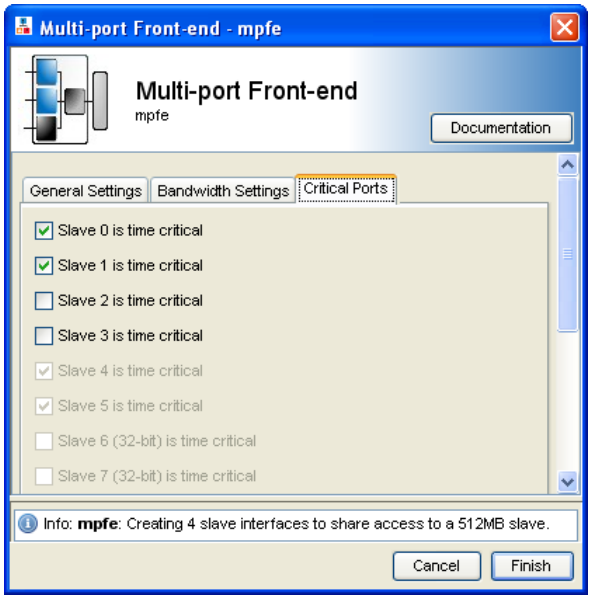
The **Bandwidth Settings** tab allows you to control the ratio of the bandwidths given to each port. By setting weights for each port, you can restrict the frequency of access allowed for each port to the external memory interface. If you assign larger numbers to a port, that port gets a larger proportion of the external memory bandwidth, while smaller numbers allow a port less bandwidth. The supported bandwidth settings are powers of two, from 1 through 512.

For example, in Figure 7, if slave 0 has a bandwidth setting of 8 and slave 1 has a setting of 4, the arbiter allocates slave 0 approximately twice the bandwidth of slave 1.

Critical Ports

Figure 8 shows the **Critical Ports** tab in the MPFE parameter editor.

Figure 8. Critical Ports Tab



The **Critical Ports** tab allows you to specify the ports in the system as time-critical or otherwise. The arbiter always grants access to the critical ports over those which are not critical. Each time-critical port gets the share of the external memory bandwidth specified on the **Bandwidth Settings** tab. When the arbiter does not receive any more requests from the time-critical ports, the non-critical ports then gets access to the external memory. The non-critical ports continue getting access until a critical port starts requesting again.

Configuring and Generating the Qsys System

To configure and generate the Qsys system, follow these steps:

1. Connect the MPFE reference design ports to your masters and memory controller. To connect the ports, follow these steps:
   a. Connect the master port of the MPFE reference design to the slave port of the memory controller.
   b. Connect the slave ports of the MPFE reference design to the master ports in your design.
   c. Connect the clock and reset output from the memory controller to the MPFE reference design and to your masters.
2. Export the appropriate conduit interfaces. The conduit interfaces include **memory_phy** and **other** on the UniPHY-based memory controller to enable connections to the external memory device.
3. Click the **System Inspector** tab to verify component properties and connections.

4. Check if there are any warning messages regarding address ranges. Select **System>Auto-Assign Base Addresses** if appropriate.
5. In the **Clock Settings** tab, specify the PLL reference clock frequency.
6. In the **Project Settings tab**, set **Limit interconnect pipeline stages** to **0** if you need to match the SOPC Builder functionality. Otherwise, use the default setting of **1** to achieve higher fabric f_{MAX}.
7. In the **Generation** tab, turn on the **Create Verilog simulation model** option and verify the output directory path.

   ☞ Currently Qsys does not support the creation of a testbench system or generic memory model in the Quartus II software version 10.1, and you must manually create a testbench for functional simulation. The design example provided with the MPFE reference design includes an example testbench and memory model for your reference.

8. Select **Generate**.

Using Standalone RTL Flow

To instantiate the MPFE reference design in a standalone design, instantiate the **mpfe_top** module in your design and add all the component RTL files to your project. However, there are a few key RTL modifications required for this design flow. To modify the RTL flow, follow these steps:

1. Use the RTL parameter settings of the **mpfe_top** module to specify the data width, address width, bandwidth and critical port settings.
2. Connect the `mst_*` signals on the MPFE reference design to the `avl_*` signals of the DDR3 memory controller. You must connect the `mst_burstcount` signal to the `avl_size` signal.
3. Invert the `avl_ready` output from the memory controller and connect it to the `mst_waitrequest` signal.

   ☞ You must invert the `avl_ready` output because the polarity of the signals is reversed.

4. Edit the **mpfe_top.v** module to convert the `mst_burst_begin` signal from an internal wire to an output port, and connect the signal to the `avl_burstbegin` port on the memory controller. When implemented in Qsys or SOPC Builder, the interconnect fabric generates the `burstbegin` signal. However, in the RTL flow, the MPFE reference design master port needs to generate the `burstbegin` signal for the memory controller.
5. Connect the MPFE reference design ports to your masters and memory controller.

Simulating the System

To simulate your system, you must create a testbench for functional simulation.

1. Instantiate the Qsys system, or top level RTL module.
2. Instantiate the memory model. You can obtain the memory model file from the memory vendor.

   Qsys currently does not generate a generic memory model. Use the vendor model, or obtain a generic model by using the MegaWizard™ Plug-In Manager flow for UniPHY-based DDR2 and DDR3 controllers.

3. Create clock and reset signals for the system.
4. Edit the Qsys-generated simulation setup script to include the testbench and memory model files, and load the testbench system.

The simulation only supports Verilog HDL language and the script generated for ModelSim®.

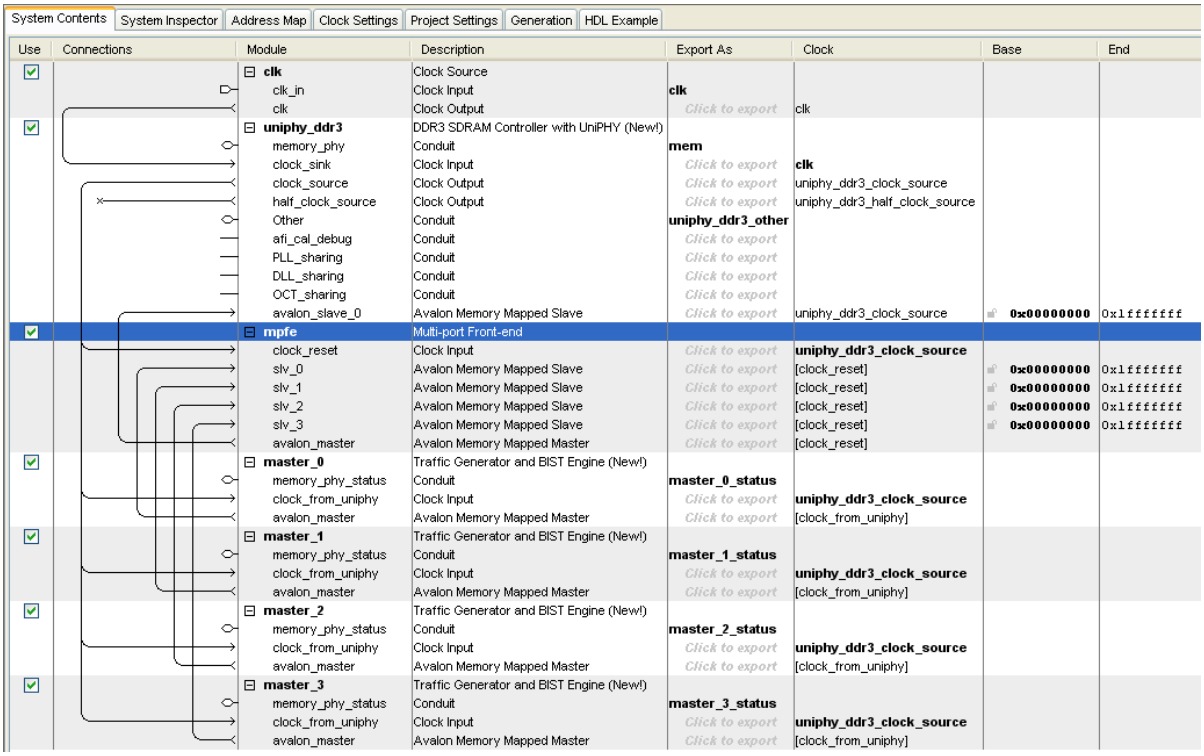
For more information, refer to the testbench file, **my_qsys_mpfe_system_tb.v**, and simulation script, **mti_setup.tcl**, provided with the design example. These files are located in the **design_example\my_qsys_mpfe_system\sim_verilog** folder.

Using the Design Example

The design example provided with this document uses the MPFE reference design to share the external external memory access between four data masters. The MPFE reference design implements the data masters using four separate instances of Altera's Traffic Generator and BIST Engine megafunction. These data masters model access patterns from different functional blocks of a typical system.

Figure 9 shows the example system contents and connections in Qsys.

Figure 9. Example System Contents and Connections in Qsys



In Figure 9, masters 0 and 1 are critical data masters in the system, and masters 2 and 3 are non-critical masters.

The bandwidth settings for masters 0 and 1 are 8 and 4, which allocate 67% of the memory bandwidth to master 0, and the remaining 33% of bandwidth to master 1. With these settings, the MPFE arbiter typically processes two requests from master 0 for each request from master 1. Because masters 2 and 3 are non-critical ports, the MPFE arbiter only processes requests from these masters when there are no pending requests from masters 0 and 1. You can observe the operation of the MPFE arbiter during functional simulation (Transcript window messages), and in the hardware (mpfe/arbiter/`arb_grant` signals).

Refer to the **README.pdf** file for more information.

You can use the design example to understand the implementation and functionality of the MPFE reference design.

To examine the implementation of the MPFE reference design, follow these steps:

1. Open the Quartus II project file in the **design_example** folder.
2. Open the **my_qsys_mpfe_system.qsys** file from the **File >Open** or **Tools >Qys** menu selection.
3. View or edit the MPFE reference design settings by selecting **mpfe** from the **System Contents** tab.

4. You can modify bandwidth settings and critical port settings, regenerate the system, and use simulation to examine MPFE functionality

To simulate the design example in ModelSim:

1. Open ModelSim and change the working directory to **\design_example\my_qsys_mpfe_system\sim_verilog**.
2. Execute the **mti_setup.tcl** script from the Transcript window.
3. Load the design, open the waveform viewer, and run the simulation by executing the following commands in the Transcript window:

```
ld
do wave.do
run -all
```

4. Examine results in the Waveform and Transcript windows.

Debug Slave Registers

Table 2 lists the registers tracking the MPFE master ports.

Table 2. Master Registers

Register	Address	Size	Description
Clear counters	0×0	32 bits	Write to this address to clear all the counters.
Master wait count	0×1	32 bits	The number of cycles that the slaves have wait-stated for the master.
Master write beats count	0×2	32 bits	The number of words of write data that the master has sent.
Master read beats count	0×3	32 bits	The number of words of read data that the master has received.

Table 3 lists registers tracking the MPFE slave ports.

Table 3. Slave Registers

Register	Address	Size	Description
Per-slave grant count	Slave offset + 0×0	32 bits	The number of times that this slave has been granted access.
Per-slave write count	Slave offset + 0×1	32 bits	The number of words of write data that this slave has sent.
Per-slave read count	Slave offset + 0×2	32 bits	The number of words of read data that this slave has received.
Per-slave worst wait	Slave offset + 0×3	10 bits	The worst number of cycles that this slave has to wait between requesting and being granted.
Per-slave total wait count	Slave offset + 0×4	32 bits	The total number of cycles that this slave has to wait between requesting being and granted.

This set of registers is the same for each slave in your system. To access the registers for any given slave, use the following formula to calculate the per-slave offset:

Per-slave offset = 0×10 + (0×8 * slave number)

Use the following formula to calculate the required address for a particular register:

Address = 0×10 + (0×8 * slave number) + register address

For example, to access the worst-case wait count (0×3) for slave 4:

Address = 0×10 + (0×8 * 4) + 0×3 = 0×33

The addresses listed are word addresses. Convert the addresses to byte addresses (multiply by 4), if you are reading them using Nios II console or System Console.

Document Revision History

Table 4 shows the revision history for this document.

Table 4. Document Revision History

Date	Version	Changes
January 2011	1.0	Initial release.