

# **MICROTRONIX**

## **AVALON MULTI-PORT FRONT END**

### **IP CORE**

---

**USER MANUAL V1.0**



Microtronix Datacom Ltd  
126-4056 Meadowbrook Drive  
London, ON, Canada N5L 1E3  
[www.microtronix.com](http://www.microtronix.com)



## Document Revision History

This user guide provides basic information about using the Microtronix **Avalon Multi-port Front End IP Core**. The following table shows the document revision history.

Date	Description
Aug 2016	First Release

### **How to Contact Microtronix**

#### **E-mail**

Sales Information: [sales@microtronix.com](mailto:sales@microtronix.com)

Support Information: [support@microtronix.com](mailto:support@microtronix.com)

#### **Website**

General Website: <http://www.microtronix.com>

FTP Upload Site: <http://microtronix.leapfile.com>

#### **Phone Numbers**

General: (001) 519-690-0091

Fax: (001) 519-690-0092

### Table of Contents

---

Document Revision History .....	2
How to Contact Microtronix .....	3
E-mail .....	3
Website .....	3
Phone Numbers .....	3
IP Core Features .....	7
Introduction .....	8
Slave Ports .....	8
Avalon Random Slave .....	8
Avalon Burst Slave .....	10
Avalon Control Port .....	11
Port Arbitration .....	12
Design Recommendations .....	13
Design Flow .....	16
Avalon Master Settings .....	17
Data Width .....	17
Address Width .....	18
Maximum Pending Reads .....	18
Add Control Port .....	18
Device Family .....	18
Scheduler Settings .....	18
Starvation control .....	18
Starvation Limit .....	18
Avalon Slave Ports .....	19
Port Type .....	19
Avalon Random Port Settings .....	19
Avalon Burst Port Settings .....	19
Port Signals .....	20
Resource Requirements .....	22
Performance .....	22
Simulation .....	22
Example Application .....	23
Installation .....	26
License .....	26

List of Figures

---

Figure 1: Random Read Transfer..... 9

Figure 2: Burst Read Transfer ..... 10

Figure 3: Qsys IP Catalog ..... 16

Figure 4: Multi-port Front-end Settings..... 17

List of Tables

---

Table 1: Control Port Register ..... 11

Table 2: Avalon Master Port Signals ..... 20

Table 3: Avalon Master Port Signals ..... 21

Table 4: Avalon Control Port Signals..... 21

Table 5: FPGA Resource Requirements..... 22

Table 6: Typical IP Core Performance ..... 22

Table 7: Burst Port 512-bit Example – Performance Summary ..... 25

Table 8: Burst Port 128-bit Example – Performance Summary ..... 25

### IP Core Features

- Advanced Performance Architecture
  - Five priority levels for port service requests
  - Up to 26 Avalon Memory-Mapped slave interfaces
  - User configurable cache for each slave port
  - Each slave port can use an independent clock
  - Two slave port types: Burst Slave and Random Slave
- Quartus II Qsys support
- Support for Altera OpenCore Plus evaluation
- Supports Max 10, Cyclone, Arria, and Stratix devices

### Introduction

The Microtronix ***Avalon Multi-port Front End (MPFE)*** provides a complete, easy-to-use solution to interface multiple Avalon Memory Mapped Masters to a single Avalon Memory Mapped Slave such as a Memory Controller.

The MPFE integrates easily into any Quartus II Qsys project. It is designed to optimize the performance of Avalon bus based video and streaming data systems.

The MPFE provides one Avalon Memory Mapped Master port and up to 26 Avalon Memory Mapped slave ports for IP components that require access to the master interface. The MPFE supports independent clocks for each slave and implements buffers in on-chip memory for each port to support high speed data transfer to the master port.

The MPFE performs arbitration between the Slave ports based on 5 service request priority levels selectable in the Qsys component.

Two types of Slave ports are available. The Burst Slave port supports standard Avalon burst transfers. The Random Slave port implements a local cache in on chip memory and can improve performance of IP components that do not support burst transfers.

Additionally, the Slave ports have data width conversion capability to interface efficiently to IP components with different data width capabilities than the master.

### Slave Ports

The Multi-Port Front End has up to twenty six Avalon Memory Mapped Slave ports. Each port has an internal FIFO, which acts as a bridge between the Slave port and the Master port. Each port can be configured as a Random Slave, or Burst Slave.

The Multi-port Front End has an optional Control Port. This port contains a register for flushing the Random Slave port caches.

**NOTE:** The FIFOs of different ports do not maintain data coherency between each other.

### ***Avalon Random Slave***

The Avalon Random Slave port is optimized for IP that does not perform burst operations. Each Random port contains a cache buffer.

The buffers are implemented as a direct mapped cache memory. On a cache hit, data is available immediately with one cycle of read latency. On a cache miss, any modified data is written back to the Avalon Master as a burst operation, and then the cache is loaded by reading



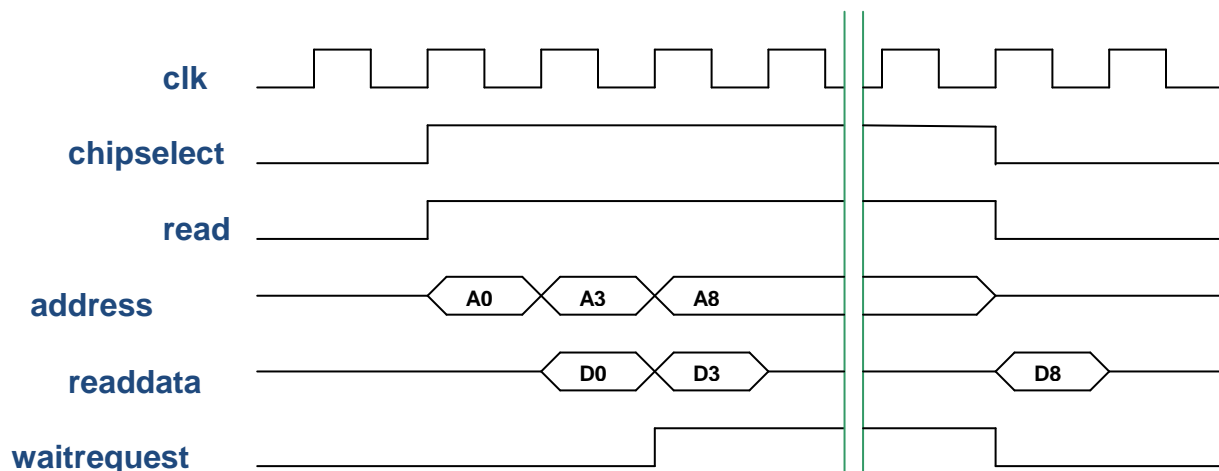
from the Avalon Master. The cache maintains full data coherency, meaning a read cycle to the last written address produces the last written data.

It is important to note that with the default settings, write data is only written to memory on a read, or a write to an address that is not in the cache. Depending on the data access patterns and the use of multiple ports, it is possible for write data to remain in the cache of a particular port for an indefinite amount of time unless action is taken to cause the data to be written.

There are two ways to cause the Random port to write out the cache. The first method is to read (or write) an address that is known to be outside the current cache, thereby forcing the cache to write out the modified data and reload from the new address. The second method of writing out the cache is accomplished by using the Control Port. Each slave port has a corresponding bit in a Control Port register that causes the cache to be written. Refer to the section **Avalon Control Port** for more information about using the Control Port.

The size of the cache can be set in the Qsys GUI. A larger cache size increases the probability of a cache hit, but also increases the waitrequest assert time and resource usage.

Figure 1 shows a Random read transfer. The cache size is set to eight words and it holds addresses A0 through A7. Address A8 results in a cache miss and the waitrequest is asserted.



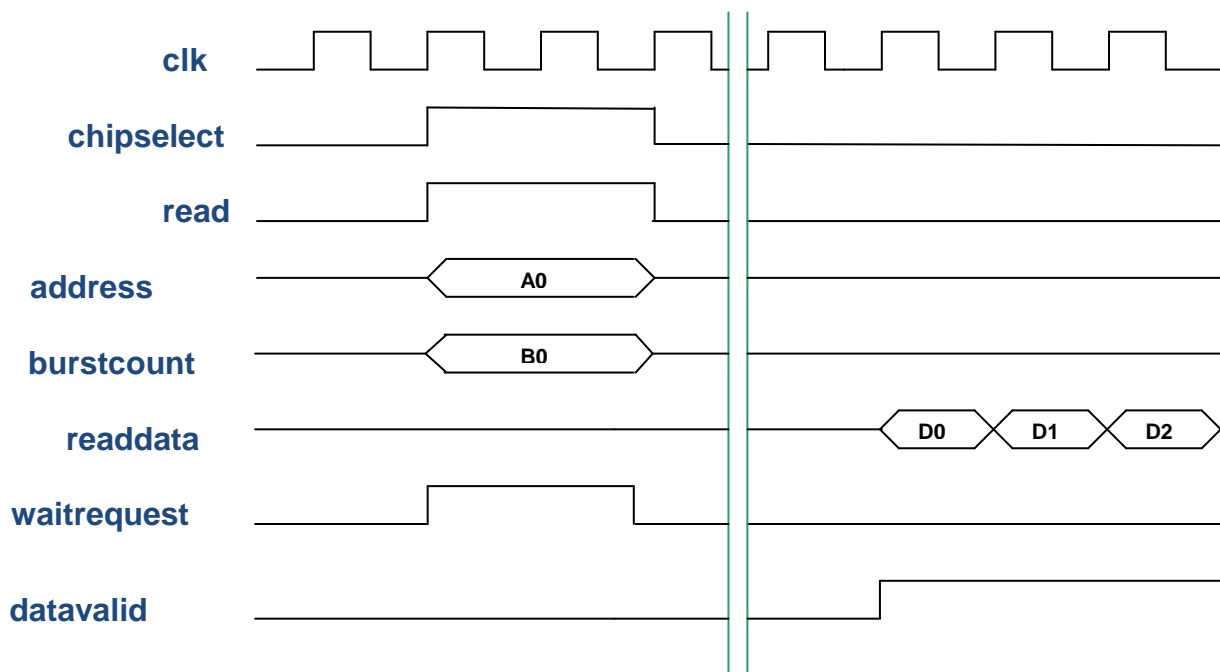
*Figure 1: Random Read Transfer*

### Avalon Burst Slave

The Avalon Burst Slave port maximizes the data throughput. It handles a transfer as a unit instead of multiple single transfers. The Burst Slave supports multiple posted write and read transfers.

Figure 2 shows a read transfer. The address and the burstcount are latched at the beginning of the transfer. As soon as the first read data is available, the datavalid is asserted. During a transfer the burst slave can de-assert the datavalid signal when no valid read data is available.

To get the most optimal read performance, the Avalon Master should post read transfers as soon as possible. It enables the Burst Slave to fetch the read data from the SDRAM memory. For more information on burst transfers see the Avalon Interface Specification document published by Altera. ([https://www.altera.com/content/dam/altera-www/global/en\\_US/pdfs/literature/manual/mnl\\_avalon\\_spec.pdf](https://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/manual/mnl_avalon_spec.pdf))



**Figure 2: Burst Read Transfer**

### Avalon Control Port

The Avalon Control Port is used (typically by Nios software) to flush the Random Slave port cache memory. The Control Port is enabled by selecting the "Add Control Port" option in the Qsys configuration GUI.

The Control Port consists of a single 32-bit, write only register at address zero. Each bit of the register corresponds to one slave port. Bit 0 (the least significant bit), controls slave port 0.

When a '1' is written to a bit in the Port Flush Register, the control port signals to the corresponding slave port to flush any unwritten data to memory. Once the port has been signaled, the bit will be cleared automatically.

If the port is configured in burst mode the flush signal has no effect because the burst port automatically presents any completed bursts to the scheduler for service.

**Table 1: Control Port Register**

Control Port Address	Description
0	<p>Port Flush Register. The bits in this register are applicable only to ports that are configured as Random type ports. When a bit is written 1, the corresponding port writes out any modified data in its cache memory.</p> <p>Bit 0 – flush slave port 0 Bit 1 – flush slave port 1 Bit 2 – flush slave port 2 ..... etc Bit 24 – flush slave port 24 Bit 25 – flush slave port 25</p>

### Port Arbitration

The scheduler offers five priority levels for slave ports. Ports of a higher priority are serviced before ports of a lower priority. The scheduler includes options to ensure lower priority ports are not starved.

The following five priority levels that are available for each slave port and can be set in the configuration GUI:

*Highest*  
*High*  
*Medium*  
*Low*  
*Lowest*

When more than one port is assigned to the same priority, requests from the slave ports are queued on a first come, first served basis as follows: For each priority level, the scheduler samples the port requests at that priority and queues a service request for each slave port that is requesting service. After all ports in the sampled requests have been added to the queued, the port requests are sampled again.

Requests with *Highest* priority are serviced first. If there are no requests with *Highest* priority, then *High* priority requests are serviced and so on. Ports having *Lowest* priority are serviced only when there are no higher priority requests. If the memory bandwidth is fully utilized by higher priority ports then it is possible that some ports may be starved (not serviced).

The optional Starvation Control parameters on the configuration GUI can be set to prevent starvation of lower priority ports. Starvation Control is available for each priority level except *Highest*. When enabled, a Starvation Limit, measured in total port service requests performed by the Multi-port Front End, is defined. When Starvation Control is enabled and the priority level has one or more ports requesting service, the scheduler counts the total number of ports serviced while that priority level is waiting for service. When the limit is reached, the priority is increased by one level and the counter resets. If the limit is reached again without a port at that priority level being serviced, the priority increases again until it reaches maximum priority.

When the priority is increased by Starvation Control, the priority boosted ports execute ahead of other ports at the new priority.

#### STARVATION CONTROL EXAMPLE:

With the following ports:

Highest Priority	– No Ports
High Priority	– No Ports
Medium Priority	– Four ports, Starvation Control Off
Low Priority	– Two ports, Starvation Control Off
Lowest Priority	– Two ports, Starvation Control On, Limit 48

If the Medium priority ports in this example are requesting memory operations at a rate that consumes all available memory bandwidth, then Low priority ports will not be serviced because Medium priority requests take precedence. Lowest priority ports however will receive service because of Starvation Control. When the Lowest priority port requests service, after 48 requests have been serviced (all of which will be from the Medium priority ports), the priority of Lowest priority ports is increased and will execute ahead of Low priority ports. However in this example, the ports will again receive no service. After another 48 requests, the Lowest priority port is granted another priority increase and executes ahead of the Medium priority ports and so one of the ports is serviced, after which the Lowest priority level loses its priority boost and the process begins again.

### Design Recommendations

#### AT LEAST TWO PORTS RECOMMENDED FOR BEST EFFICIENCY

The Multi-port Front End is intended for designs with at least two ports accessing the master. The maximum data throughput of a single active slave port may not allow the full bandwidth of the master to be utilized.

#### SHARE PORTS TO REDUCE RESOURCE USAGE

It is possible to connect each IP component to a separate slave port of the Multi-port Front End, but to minimize resource usage it is generally better to share the ports. For example in a design with three frame buffers, the three write master interfaces can be connected to a single port on the Multi-port Front End, and the three read master interfaces can connect to another slave port on the Multi-port Front End, assuming they use the same clock and have the same priority requirements.

#### MEMORY BLOCK USAGE

On-chip memory blocks can be configured only into specific width and depth configurations as shown in the Altera Device Handbooks. The maximum width of a block may be 40 bits for example. Therefore when used as a buffer or cache for a wide data bus, a minimum number of memory blocks will always be required even with very small buffer sizes. For the slave port buffer sizes that are typically used with the Multi-port Front End, this minimum number of blocks often determines the number of blocks used for the port.

For Burst ports, it is most efficient to select a buffer size that makes use of as many of the bits in the memory blocks as possible. There is no benefit in making the buffer small and leaving unused bits in the memory blocks that have been allocated.

For Random ports, it may be more efficient to use a smaller cache to avoid the requirement to load a large amount of data on each cache miss. The optimal cache size depends on data access patterns.

### BURST SIZE

In almost all cases, using small burst sizes lead to less efficient operation. Consider increasing the burst size to improve data throughput.

### RANDOM PORT RESOURCE USAGE

When using the Random Slave ports, cache tag registers are also required. For a wide bus and a large cache, the tag registers can cause high logic cell usage by the port. In this case it may be preferable to reduce the cache size.

### BYTE ENABLE FOR PORTS WITH REDUCED DATA WIDTH

Burst ports include the option of reducing the data width to be less than that of the Avalon Master. This option may be used to reduce interconnect resource usage for ports that require reduced bandwidth, but restrictions on how the port is used must be observed to prevent the generation of byte enable patterns that don't conform to the Avalon Interface Specifications.

The Avalon Interface Specification supports only certain patterns of the byte enable bits. All selected bits must be sequential, the number of bits selected must be a power of 2, and the selected bits must be aligned with respect to the selected data width.

When operating at a reduced width, a port will combine 2, 4, or 8 words from the slave port into a single word on the Avalon Master port for slave port write operations. Depending on the alignment and length of data accesses on the slave port and the usage of the byte enable signal, the resulting combined data word may not conform to the supported byte enable combinations.

To avoid generating unsupported byte enable values, write to the reduced width slave port with bursts that start at an address aligned to a word no smaller than  $\frac{1}{2}$  the width of the Avalon Master, set the burst length to avoid writing partial words to the Avalon Master, and disable the byte enable signals for the port (or drive all bits high).

The MPFE will function if these recommendations are not followed, but the component(s) that are attached to the Avalon Master port may not support the byte enable signals correctly.

There are no restrictions on the use of a slave port that is configured for double the width of the Avalon Master, or when reading from a reduced width slave port.

### MISALIGNED OPERATIONS ON PORTS WITH REDUCED DATA WIDTH

When writing to a burst slave ports with a data width less than the Avalon master's data width, the port combines slave data words to generate data words with the master's width and aligned to the master word addresses.

If a write to the slave port that is not aligned to the Avalon Master's data words, the MPFE must generate the additional words needed to align the write. This process requires one clock cycle per word, during which time the wait request signal is asserted.

For the case of small misaligned bursts accesses to ports with a large width reduction factor, the alignment can significantly lower the efficiency of writing data to the slave port.

### ASSIGN PORTS SEQUENTIALLY FOR LOWEST RESOURCE USAGE

The ability to arbitrarily disable any slave port of the Multi-port Front End may be useful in developing and debugging designs, but may increase resource use in some cases. For minimum resource usage it is recommended to use a sequential range of ports starting at port 0.

### BURST SPLITTING

The Multi-port Front End may split burst requests from the slave ports for various reasons. Avalon slave reads may be split if the buffer does not have free space for the entire burst, either because the buffer is smaller than the burst size, or the buffer still contains data from a previous burst that has not yet been returned to the slave. Read burst splitting is more likely to occur when the slave port is narrower than the master, or has a slower clock because the master supplies data faster than it can be returned to the slave. Avalon slave writes larger than half the buffer size will be split.

### CONFIGURING PORTS FOR A NIOS II PROCESSOR

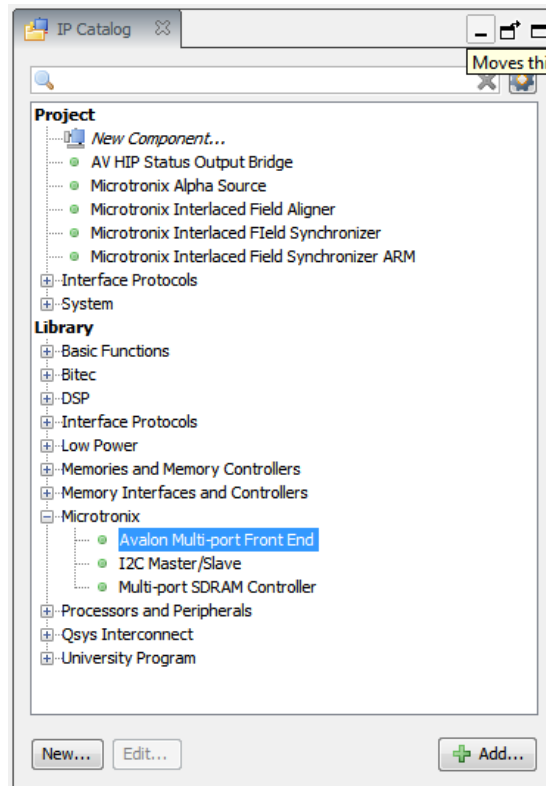
To maximize Nios II processor performance, it is recommended to configure the processor to use instruction and data caches, enable burst transfers for both, and to connect both ports to a single burst port on the MPFE. The MPFE port should support the Byte Enable signal and have a data width ratio setting of 1 or 2.

The NIOS II is also compatible with the Random port type. A 32 byte buffer is recommended when a Random port is used. Larger buffers typically result in reduced performance.

### Design Flow

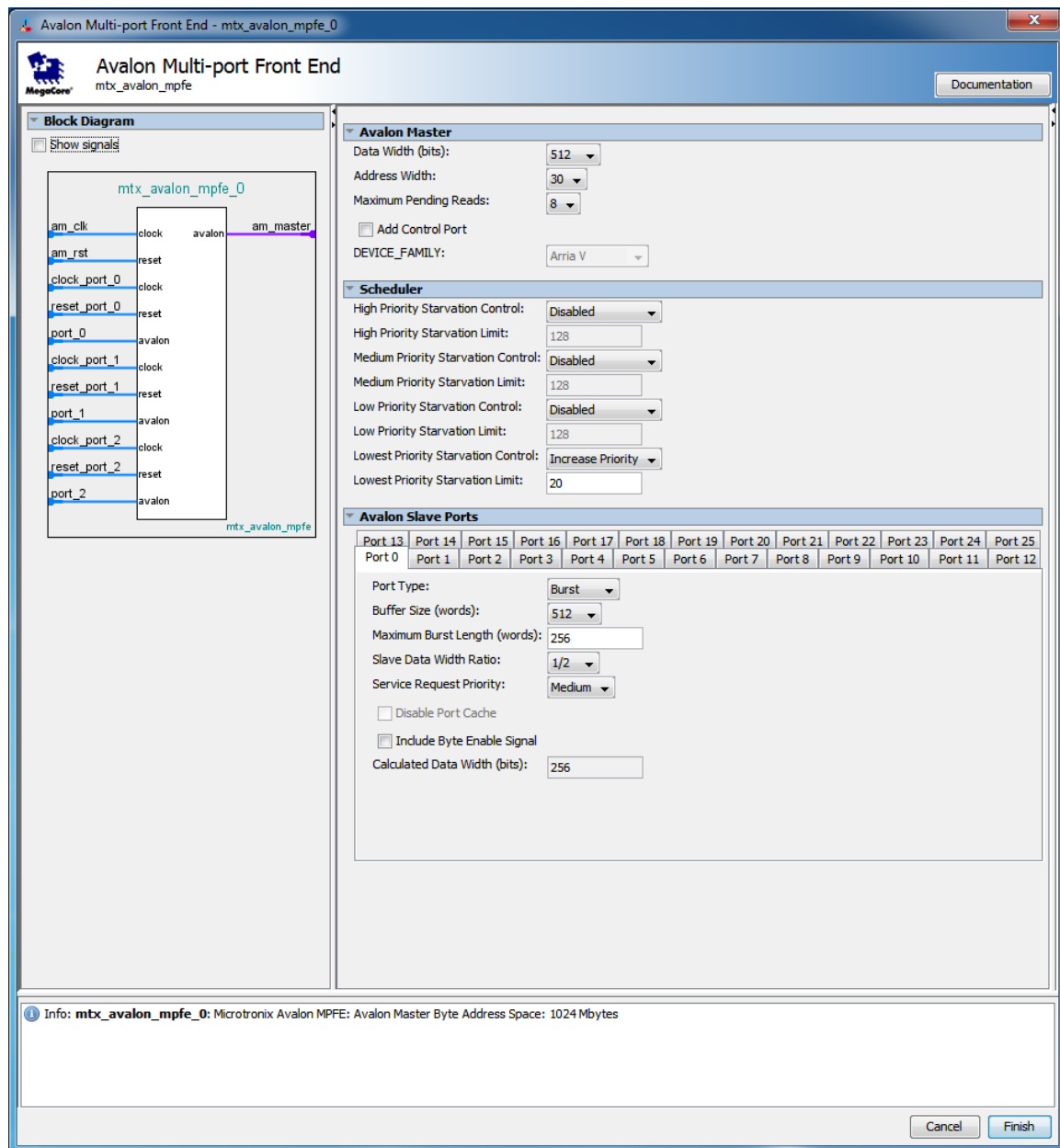
The following steps describe how to integrate the Multi-port Front End in a Qsys system.

- Create or open a Quartus project.
- Launch Qsys.
- Build your system by adding components from the IP Catalog list.
- Choose the Avalon Multi-port Front End from Library > Microtronix, then click Add.



**Figure 3: Qsys IP Catalog**





**Figure 4: Multi-port Front-end Settings**

### Avalon Master Settings

The Avalon Master section of the configuration screen sets the properties of the Avalon master port.

#### Data Width

This specifies the data width of the Avalon master port. It should be set to match the width of the port the master connects to so that Qsys does not generate a data width adapter between the two components.

### ***Address Width***

This setting specifies the number of address bits for the Avalon master port. The number of bits is specified in terms of byte addresses. Select the number of bits required to address the memory. For example for a 1 GB memory, 30 bits are needed. The supported address range is from 8 bits (256 bytes) to 40 bits (1024 GB)

### ***Maximum Pending Reads***

This setting specifies number of pending reads the Avalon master port of the Multi-port Front-end may request. The setting is a maximum limit. Depending on the slave port requests and the order in which they are serviced, the actual number of pending reads may be less than the limit.

### ***Add Control Port***

When this option is enabled, a Control Port is added to the Multi-port Front End. The Control Port allows the cache memory of the Random Slave ports to be cleared.

### ***Device Family***

The selected Altera device family is displayed for reference

## **Scheduler Settings**

The Scheduler settings section contains that affect the operation of the Scheduler that selects slave ports for service.

### ***Starvation control***

The Starvation Control setting is provided for all port priority levels except *Highest* priority. Its purpose is to ensure that lower priority ports will still be serviced at a controllable rate if higher priority ports are consuming all memory bandwidth in the system.

When set to *Disabled*, the ports assigned to the corresponding priority level are serviced only if there are no higher priority ports requesting service.

When enabled, the scheduler will increase the priority of requests from ports at the corresponding priority by one level after a specified number of ports have been serviced without a port at this priority level receiving service.

### ***Starvation Limit***

These settings are enabled only when Starvation Control for the corresponding priority has been enabled. The parameter sets how many higher priority ports can be serviced before the scheduler increases the priority of ports at this priority level.

### Avalon Slave Ports

This section enables and configures the Avalon slave ports. Each of the 26 possible ports has a separate configuration tab.

#### *Port Type*

This enables ports and controls the port type. Ports can be configured as Avalon-MM Random or Avalon-MM Burst slaves, or they can be disabled.

It should be noted that no memory protection or management schemes are provided. If needed, a master peripheral connected to the local Avalon port must take care that data is only written in its own address space and no data is overwritten in the other address spaces.

#### *Avalon Random Port Settings*

##### **BUFFER SIZE**

This specifies the size of the port's internal buffer in Avalon words. For the best Nios II processor performance, the buffer size should be limited to 8 or less words.

##### **DOUBLE THE AVALON DATA WIDTH**

Enabling this option will set the Avalon data width to twice the standard data width. Best performance is achieved when the port's data width matches the width of the Avalon masters connected to it, where possible.

##### **FORCE BUFFER RELOAD ON EVERY AVALON READ**

Enabling this option will disable any read caching. Whenever an Avalon master reads from this port, it will fill its internal buffer from memory.

**NOTE:** This setting will significantly decrease performance

#### *Avalon Burst Port Settings*

##### **BUFFER SIZE**

This specifies the size of the port's internal FIFOs in Avalon words. There are two FIFOs in a burst port, one for reads and one for writes.

##### **MAX BURST LENGTH**

This specifies the maximum length of Avalon burst transfers to this port. This value should be at least as large as the maximum burst size of any connected Avalon masters.

##### **DATA WIDTH RATIO**

The data width of the slave port can be set in terms of the Avalon master data width. The choices available are 2, 1,  $\frac{1}{2}$ ,  $\frac{1}{4}$ ,  $\frac{1}{8}$  and  $\frac{1}{16}$  of

the width of the master. For example if the master is 256 bits and the  $\frac{1}{2}$  option is chosen, the slave port will be 128 bits wide. Refer to the Design Recommendations section for more information about using reduced data width slave ports.

### INCLUDE BYTE ENABLE SIGNAL

The byte enable signals are included in the Avalon slave interface by default, but can be disabled by unchecking this selection. When byte enable is not present, all bytes in each data word are written to memory on an Avalon slave write operation.

If the byte enable signal is enabled for burst ports with data width less than the Avalon master, a Qsys warning message is generated because, depending on how the byte enable signals are used, the resulting byte enable bit pattern on the Avalon master may not conform to the Avalon Bus specifications.

## Port Signals

**Table 2: Avalon Master Port Signals**

Signal	Direction	Description
am_clk	Input	Clock input for the Avalon master port.
am_rst	Input	Reset input for the Avalon master port.
am_addr[]	Output	Address output of the Avalon master port. The width of the address signal is set in the MPFE configuration panel by the Avalon Master's "Address Width parameter". Address widths from 8 to 40 bits are supported.
am_data_in[]	Input	Data input for the Avalon master port. The data width is set in the MPFE configuration panel by the Avalon Master's "Data Width" parameter. The width should be set to match the slave that is connected to the Avalon master port. Widths of 16, 32, 64, 128, 256, 512, and 1024 bits are supported.
am_data_out[]	Output	Data output from the Avalon master port. The width is always the same as the am_data_in[] signal.
am_wr	Output	Write output from the Avalon master port.
am_rd	Output	Read output from the Avalon master port.
am_waitreq	Input	Wait request input.
am_be[]	Output	Byte enable output from the MPFE. The width of this signal equal to the Avalon master data width / 8.
am_burstcount[]	Output	Burst width output from the Avalon master port. The width of this signal is determined from the settings of the individual slave ports.
am_burststart	Output	This signal is asserted at the start of a burst on Avalon master port.
am_datavalid	Input	Data valid input for reads by the Avalon master port.

**Table 3: Avalon Master Port Signals**

Signal	Direction	Description
as_port_N_clk	Input	Clock input for the slave port. The clock may be the same as the Avalon master clock, or a different clock may be used.
as_port_N_rst	Input	Reset input for the slave port.
as_port_N_addr[]	Input	Address input for the slave port. The width of this signal depends on the Data Width Ratio parameter of the slave port and the width of the Avalon master port.
as_port_N_data_in[]	Input	Data input for the slave port. The data width depends on the data width of the Avalon master port and the Data Width Ratio parameter of the slave port.
as_port_N_data_out[]	Output	Data output of the slave port. The data width depends on the data width of the Avalon master port and the Data Width Ratio parameter of the slave port.
as_port_N_wr	Input	Write input to the slave port.
as_port_N_rd	Input	Read input to the slave port.
as_port_N_cs	Input	Chip select input of the slave port.
as_port_N_waitreq	Output	Wait request output from the slave port.
as_port_N_be[]	Input	Byte Enable inputs. This signal is present when the "Include Byte Enable Signal" of the slave port is selected. When the byte enable signal is not present, a write operation to the port will write all bytes.
as_port_N_burstcount[]	Input	Burst count input to the slave port. This signal is present only on the Burst type slave port. The width of the signal is determined by the "Maximum Burst Length" parameter of the slave port.
as_port_N_datavalid	Output	Data valid input for the slave port.

**Table 4: Avalon Control Port Signals**

Signal	Direction	Description
as_ctrl_clk	Input	Control port clock input.
as_ctrl_rst	Input	Control port reset input.
as_ctrl_addr[5..0]	Input	Address input for the control port.
as_ctrl_data_in[31..0]	Input	32 bit data input used to write control port registers.
as_ctrl_data_out[31..0]	Output	32 bit data output used to read the control port registers. (For future use)
as_ctrl_rd	Input	Read signal for the control port.
as_ctrl_wr	Input	Write signal for the control port.
as_ctrl_waitreq	Output	Wait Request for control port operations.

### Resource Requirements

Table 5 shows an example of resource usage in logic elements (LE) and memory blocks for the various modules of the Avalon MPFE IP Core. This example is based on a Arria V device and MPFE configured with 1 random slave port and four burst slave ports with the Avalon master port set for a 512 bit data width. The burst slaves have a data width ratio of  $\frac{1}{4}$  and the random port has a data width ratio of 1. The burst ports have a 512 word buffer and the burst port has a 32 word cache. Two scheduler priority levels are used. LE's are calculated based on 2.65 LE per ALM. All slave ports are connected to masters that support both read and write operations.

**Table 5: FPGA Resource Requirements**

Module	LE*	M10K RAM Blocks*
Arbitration Controller	3983	15
Avalon Random Port	240	32
Avalon Burst Port	1201	29

**\*NOTE:** The actual number of logic elements and memory blocks used will vary depending on the device family, Quartus II settings, and MPFE settings including the data and address widths, buffer sizes and scheduler settings. Resource usage is reduced if some slave ports connect to read only or write only masters.

### Performance

Table 6 shows the maximum performance results for the MPFE IP Core. Actual performance may be affected by the memory width, system LE count and the FPGA pin assignments.

**Table 6: Typical IP Core Performance**

Device	Speed Grade (Commercial)	System Fmax (MHz)
MAX 10	-7	120 MHz
Cyclone V	-6	160 MHz
Arria V	-5	170 MHz
Stratix V	-2	300 MHz

### Simulation

A precompiled simulation library is provided for performing simulations using ModelSim. The library is located in the <install\_dir>/simulation directory. Perform the following steps to simulate your design with the Multi-port Front End.

1. Launch ModelSim
2. Map the Multi-port Front End library. At the ModelSim prompt type;  

```
vmap mtx_mpfe <install_dir>/simulation/mtx_mpfe
```

If you use a newer version of ModelSim, you must refresh the precompiled library. At the Modelsim prompt type;

```
vcom -refresh -work mtx_mpfe
```
3. Compile the sdram top level. Eg. If the Multi-port Front End was named `mpfe` in the SOPC Builder, then type;  

```
vcom -93 mpfe.vhd
```
4. Compile all of the design files
5. Start the ModelSim simulation by typing;  

```
vsim -t ps -L mtx_mpfe <top_level>
```

This procedure assumes all of Altera's simulation libraries are installed and available to ModelSim. If this is not the case, use the following procedure to generate the libraries with Quartus.

1. From the Tools menu, select Launch EDA Simulation Library Compiler.
2. Select ModelSim from the Tool name list. Browse to and select the location of the ModelSim executable.
3. Select the appropriate device family.
4. Select the desired language and output directory.
5. Click Start Compilation.
6. Once the libraries are generated, they must be mapped into ModelSim.

### Example Application

An example video design has two 3G-SDI inputs and two externally generated overlay layers with RGB and Alpha planes, one overlay is supplied from an LVDS video interface, and the other from a parallel display interface. All input sources are connected to a VIP Alpha Blending Mixer component to generate a 3G-SDI output.

Each SDI video input has a clipper and scaler in the video path, allowing each to be cropped, zoomed, and scaled. The mixer allows the input video sources to be displayed full screen, or as reduced size windows located at a programmable location in the output video. The overlay layers are mixed on top of the two SDI inputs.

The FPGA is an Arria V GX with a 64 bit bank of memory clocked at 475 MHz using a Uniphy quarter rate soft memory controller. The memory controller clock `afi_clk` is 118.75 MHz, too slow to be used as the VIP clock for 3G. Therefore a 140 MHz clock is used for the VIP components and the "Use separate clocks for the Avalon-MM master

interfaces" option of the frame buffers is enabled, with the Avalon master clocks driven by the memory controllers `afi_clk` output.

The design contains 6 frame buffers. Maximum memory bandwidth is required when both SDI inputs are full size (not scaled down or cropped). The video line average bandwidth is calculated because the Clock Video Input components have a FIFO large enough to average the bandwidth usage over the active and inactive pixels of each line. The term `Eff0` is  $(256/240)$  and represents the packing inefficiency of 20 bits pixels stored in 256 bit wide data words.

For each of the four frame buffers in the SDI video paths:

SDI Frame Buffer:

Input Side:  $(1920 \text{ active pixels} / 2200 \text{ pixels per line}) * (148.50 \text{ MHz}) * (20 \text{ bits}) * \text{Eff0} = 2.765 \text{ Gbps}$

Output Side:  $(1920 \text{ active pixels} / 2200 \text{ pixels per line}) * (148.50 \text{ MHz}) * (20 \text{ bits}) * \text{Eff0} = 2.765 \text{ Gbps}$

For each of the two overlay inputs

Overlay 1, Frame Buffer A,

Input Side:  $(1920 \text{ active pixels}) * (27440 \text{ Hz Horizontal clock}) * (32 \text{ bits}) = 1.686 \text{ Gbps}$

Output Side:  $(1920 \text{ active pixels} / 2200 \text{ pixels per line}) * (148.50 \text{ MHz}) * (32 \text{ bits}) = 4.147 \text{ Gbps}$

Total:  $4 * (2.765 + 2.765) + 2 * (1.686 + 4.147) = 33.768 \text{ Gbps}$

The memory can be conservatively expected to provide a usable bandwidth of:

$$(64 \text{ bits}) * (475 \text{ MHz}) * 2 * (70\% \text{ eff}) = 42.56 \text{ Gbps}$$

The estimated available memory bandwidth exceeds the required value by 8.8 Gbps and the design is expected to function. However when the design is tested, it is found to work with HD-SDI video inputs, or a single 3G-SDI input, but when two 3G-SDI inputs are connected, the video is skewed or too unstable for the monitor to display.

This design can be improved by the addition of the Multi-port Front End.

One reason this design doesn't make efficient use of the memory bandwidth is that the frame buffers used in this design have a maximum data bus width of 256 bits. The width difference reduces the efficiency of data transfer between the frame buffer ports and the memory controller. The Microtronix MPFE can improve the design because each burst slave port has an on-chip memory buffer. All the slave ports on the MPFE can simultaneously write to or reading from their on-chip buffers. When a port has a burst ready for service and is granted access to the memory controller bus by the scheduler, the port accesses the memory controllers across a 512 bit data bus, making the most efficient use of the Avalon bus data transfer capabilities.

Additionally, the scheduling algorithm of the Multi-port Front End reduces the maximum time a port might wait for service as compared to a round robin scheduler.

Adding the MPFE to the design with 4 burst slave ports (512 bits wide) and connecting three frame buffer Avalon Masters to each port results



in a more efficient design that operates correctly with 3G-SDI video connected to both inputs. The performance increase comes at the cost of additional logic and memory resources in the FPGA:

**Table 7: Burst Port 512-bit Example – Performance Summary**

Resource	Original Design Without MPFE	Design with MPFE configured with four 512-bit slave ports
Logic utilization (in ALMs)	34,936 / 58,900 ( 59 % )	37,493 / 58,900 ( 64 % )
Total registers	65937	70293
Total block memory bits	4,258,360 / 10,762,240 ( 40 % )	4,802,980 / 10,762,240 ( 45 % )
Total RAM Blocks	721 / 1,051 ( 69 % )	807 / 1,051 ( 77 % )

As can be seen, the addition of the MPFE used an additional 2557 ALMs and 86 RAM blocks. However, there are more efficient ways that the Multi-port Front End can be utilized in this design.

By using additional options of the MPFE the design can be optimized. If the width of each slave port is changed to 128 bits, and each frame buffer is reconfigured to have a 128 bit wide Avalon MM interface, the resource usage is reduced:

**Table 8: Burst Port 128-bit Example – Performance Summary**

Resource	Original Design without MPFE	Design with MPFE configured with four 128-bit slave ports
Logic utilization (in ALMs)	34,936 / 58,900 ( 59 % )	33,223 / 58,900 ( 56 % )
Total registers	65937	63334
Total block memory bits	4,258,360 / 10,762,240 ( 40 % )	4,100,512 / 10,762,240 ( 38 % )
Total RAM Blocks	721 / 1,051 ( 69 % )	666 / 1,051 ( 63 % )

In this configuration, it can be seen that the resources saved by using a narrower Avalon MM Interface to the frame buffers is larger than the resources consumed by the MPFE, resulting in a net reduction in ALMs and RAM blocks. In this configuration, the performance of the design was increased and total resources required were reduced by the addition of the MPFE.

### Installation

Follow these steps to install the Microtronix Multi-port Front End on your computer.

1. Insert the Microtronix Multi-port Front End Installation CD into your CD-ROM (or equivalent)
2. The setup program for the package should start. If it doesn't, browse to the CD using Windows Explorer and double-click on the setup icon.
3. Follow all the prompts. The setup program will attempt to auto-detect the installation location of the Quartus II and Nios II Embedded Processor. Please correct the specified paths if the setup program doesn't or incorrectly detects them.

### License

A valid IP core License is required from Microtronix to generate program files incorporating the Multi-port Front End IP-core. These licenses are generated based on a NIC or Guard ID supplied by the user. A Node Locked License is tied to the NIC ID of a PC or workstation and supports a single user. A Floating Server License is installed on a Windows or Linux server and supports 1,2, 3 or 5 simultaneous (floating) users.

After purchasing a license you receive your license file. Copy the license file (license.dat) to your current Quartus license file and the Multi-port Front End (CC21\_62xx) will show in the Quartus License Setup.

To support customer trial evaluations of the core in real hardware, Microtronix offers short term Evaluation Licenses using Altera's OpenCores Plus license feature. This License enables users to build sof files to run the core in-circuit for a time limited period.

Please contact Microtronix sales ([sales@microtronix.com](mailto:sales@microtronix.com)) for licensing details.