AccelDSP Synthesis Tool

Release Notes

UG635 (v11.1) April 27, 2009

XILINX®

Table of Contents

Release 11.1

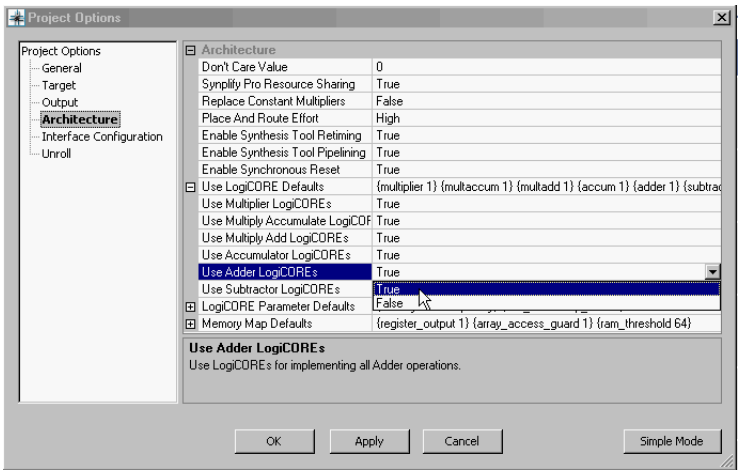
AccelDSP Enhancements

New LogiCORE Support for Increased Performance

Adder/Subtractor

With this release, the AccelDSP synthesis tool will now infer the Adder/Subtractor LogiCORE for VHDL flows.

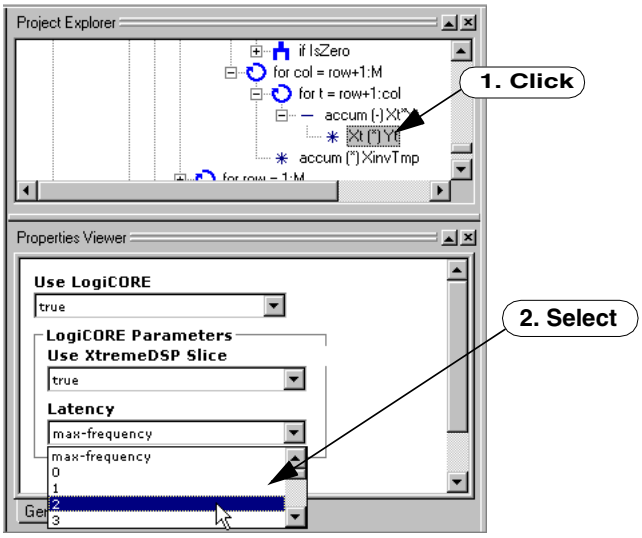
As shown below, this auto inference is not turned on by default, so you must change the **Use LogiCOREs Default** project option to turn this feature on.



New GUI Support for LogiCORE Parameters

For VHDL flows, inferred LogiCORE parameters can now be changed from the AccelDSP GUI. As shown below, simply select the associated operator from the Project Explorer

window or the Fixed Point Report, then change a parameter (like Latency) in the Properties Viewer window.



Global/Project Options

The following features have been added to AccelDSP and may be specified as Project Options or Global Options.

"Use LogiCORE Defaults" Global/Project Option Now On by Default

The **Use LogiCORE Defaults** project option is now turned on by default for VHDL flows. This causes AccelDSP to automatically infer a LogiCORE for each supported operator except the Adder and Subtractor. The default setting are as follows:

```
SetProjectOption -use_logicores {accum 1} {adder 0} {multaccum 1}
{multadd 1} {multiplier 1} {subtractor 0}

SetGlobalOption -use_logicores {accum 1} {adder 0} {multaccum 1}
{multadd 1} {multiplier 1} {subtractor 0}
```

You can turn on the use of the adder and the subtractor by setting the associated value to 1 (true).

This global/project option specifies the use of a LogiCORE for a particular type of operator like multiplier. You can turn off the use of a LogiCORE for any individual multiplier with a **Use LogiCORE** directive. For example:

```
SetDirective {for n.design.for m.a1 (*) in1} -use_logicore 0
```

New "LogiCORE Parameter Defaults" Global/Project Option

A new project option called **LogiCORE Parameter Defaults** has been created. This allows you to set LogiCORE parameters for latency and the mapping to the XtremeDSP Slice (which is the DSP48 element in the target technology). The default settings are as follows:

```
SetProjectOption -logicore_defaults {latency max-frequency}
{use_xtremedsp_slice 1}

SetGlobalOption -logicore_defaults {latency max-frequency}
{use_xtremedsp_slice 1}
```

Enhanced Register Inputs/Outputs Option for Higher Performance

In prior releases, the inputs to your design could be registered with one register and the outputs were registered with one implicit register by default. Now this feature has been enhanced to allow additional registers on the inputs and outputs.

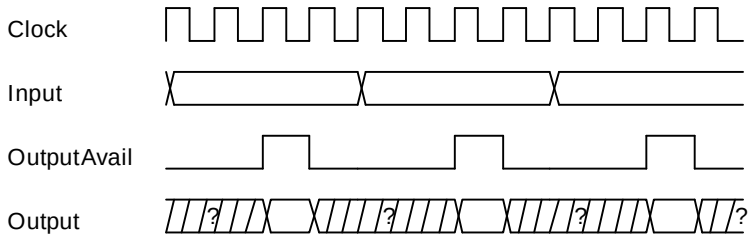
For example, assume that the critical path includes a long route from other hardware to the input of your design. By adding another register to the input, the register can be used by the place & route software to break the critical path. The project option setting might look like the following:

```
SetProjectOption -register_inputs 2
```

Now assume that you want to hold the output of your design longer to give the downstream hardware more time to capture the data. The command to add another register to the output might look like the following:

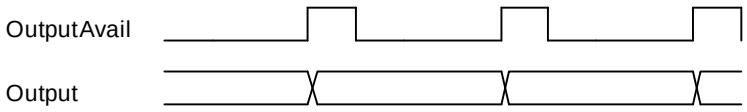
```
SetProjectOption -register_outputs 2
```

The figure below illustrates the timing diagram of a typical push-mode design with one implicit output register (the default). Notice that the output is valid only when the **OutputAvail** signal goes true for one clock cycle.



SetProjectOption –register_outputs 1

The figure below illustrates the interface timing when the project option **Register Outputs** is set to 2:



SetProjectOption –register_outputs 2

Notice that the output is valid now when the **OutputAvail** signal goes true and remains valid until the **OutputAvail** signal goes true again. This may be for more than one clock cycle. Although this adds additional cycles of latency, it gives the down stream hardware more time to capture the output data.

Note: Double registering the output is only valid for constant throughput designs with a push mode interface.

New Project Option “RAM Threshold”

A new Global/Project Option called **RAM Threshold** has been added to allow the automatic mapping of array variables to random access memory. The default is 64, so every array variable in your design that is equal to or greater that 64 elements will be automatically mapped to RAM. The benefit of mapping to RAM is reduced area and improved Fmax. You can turn off the automatic mapping to RAM by specifying a RAM Threshold of 0 (zero).

Directives

Memory Map Options “sp_sync_ram” and “sp_sync_rom” are Deprecated

Starting with this release, the memory map options **sp_sync_ram** and **sp_sync_rom** have been deprecated. If you have a current design that specifies one of these options, the option will be automatically replaced with **dp_sync_ram**.

New Memory Map Option “Array Access Guard”

AccelDSP has a new feature that allows you to request more aggressive scheduling for arrays in feedback loops. This feature is called **Array Access Guard** and is turned on (True) by default. When the Array Access Guard is on, AccelDSP does not pipeline feedback loops as aggressively.

If your design has array variables that are accessed in a known manner, but are not statically deterministic via the MATLAB code alone, and you want to improve the thoughput of the design, you can turn off the Array Access Guard on one or more array variables, then re-run the AccelDSP flow to see if the throughput improves.

If you receive an error message (E-VERIFY-0007) during the Verify RTL step, then the act of removing one or more Array Access Guards is causing a conflict.

The variable(s) with conflicts are identified in the Verify RTL step of the AccelDSP log file (accel.log). To resolve the conflict, you must re-apply the Array Access Guard to the offending variable(s).

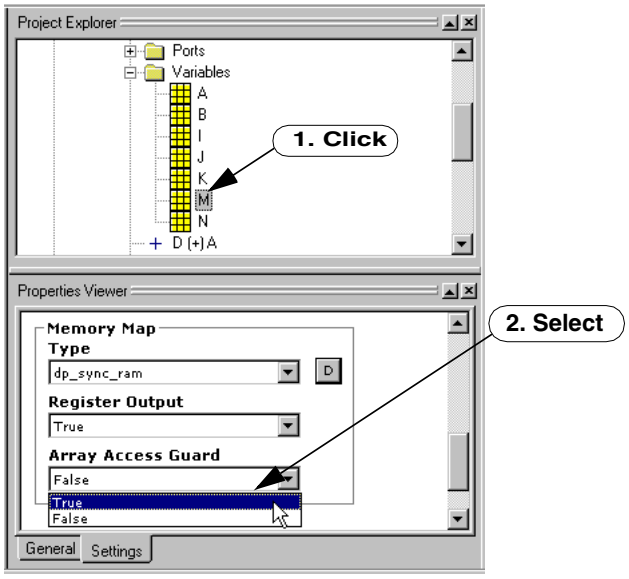
As an example, assume that you have several array variables in your design and you disable the Array Access Guard on each one to allow AccelDSP to more aggressively pipeline the design. You then re-run the AccelDSP flow and see the following message after the Verify RTL step:

```
#( E-VERIFY-0007): Simulating VHDL design: Modelsim Failed.  This is due
to a pipeline register in a feedback loop.
# Remedy: Enable the Array Access Guard for feedback variable(s)
indicated in the accel_verify.log file and rerun Generate -fixedpoint.
```

Following the directions in the message, you scroll up in the Transcript window and see the following message from the ISE Simulator:

```
# ** Warning: Encountered array conflict - Please enable the Array
Access Guard for variable 'M'.
```

To remedy the conflict, you then select the variable M in the Project Explorer window, as shown below, and set the Array Access Guard in the Properties Viewer window back to **True**.



Generate Fixed Point Report Enhancements Aid in Achieving Higher Performance

Variables List Report

The figure below shows the new Variables List tab within the Generate Fixed-Point Report.

FixedPointReport.htm

Generate Fixed Point Report

Elapsed Time: 14.12 seconds

Only one variable mapped to RAM

Variables by Hierarchy | Variables List | Operators List | Loops List | Functions List

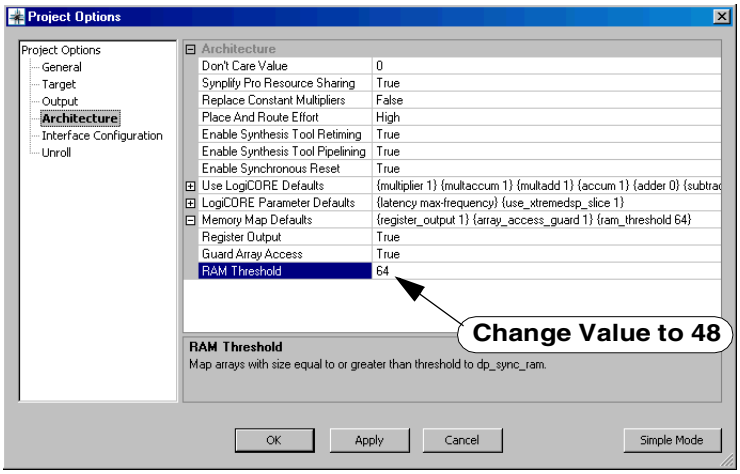
Variable	Shape	Elements ↑	Quantizer	Quantizer Source	Bits	Memory Map	Memory Map Source	Registered Output	Type
tap_delay	2 x 48	96	fixed floor wrap [16 0]	Auto Inferred	1536	dp_sync_ram	Automatically Inferred	Yes - Project Option	Double Persiste
DelayLine	3 x 20	60	fixed floor wrap [16 15]	MATLAB Source	960	nomap	Project Option	Yes - Project Option	Double Persiste
DelayLine	3 x 20	60	fixed floor wrap [16 15]	MATLAB Source	960	nomap	Project Option	Yes - Project Option	Double Persiste
CT	1 x 53	53	ufixed floor saturate [52 51]	**Hierarchical Directive**	2756	nomap	Project Option	Yes - Project Option	Double Consta
coeff	48 x 1	48	fixed floor wrap [16 15]	MATLAB Source	768	nomap	Project Option	Yes - Project Option	Double Persiste Consta

Figure 1-1: **Variables List Fixed Point Report**

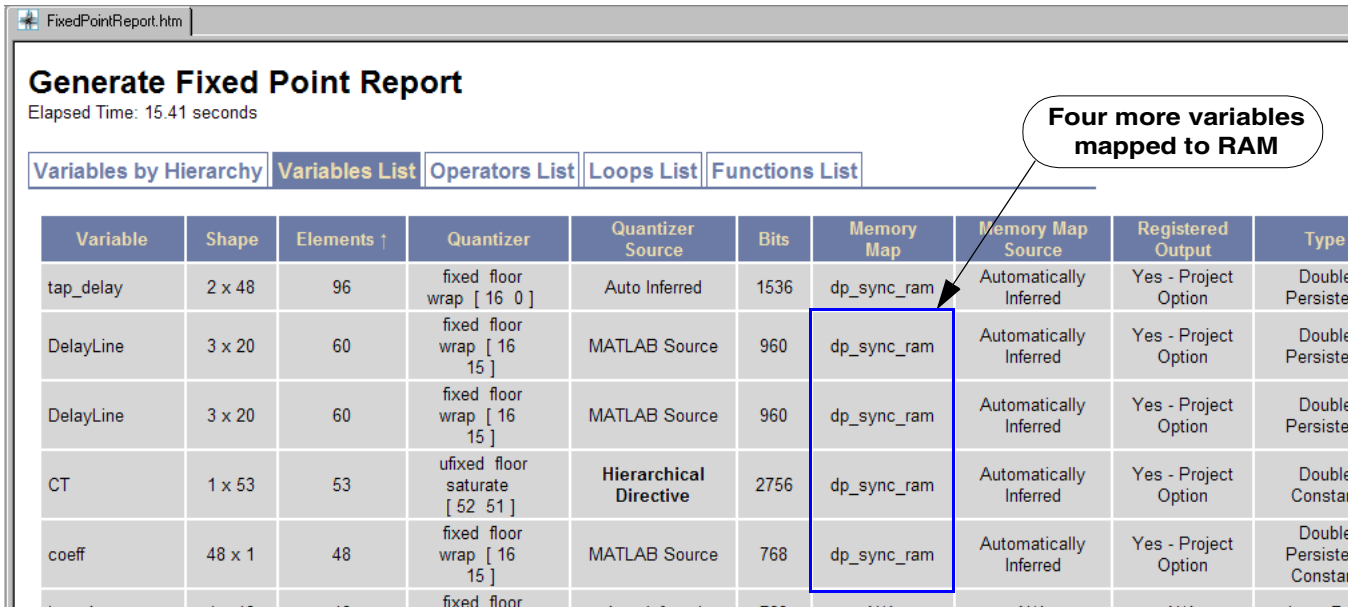
This report lists each design variable and provides important information about the variable attributes. You can change an attribute by clicking on the variable row, where you

can change the attribute value in the Properties Viewer. In this example, you can see that only one array variable is mapped to Dual-Port RAM.

You can use this report to tune your design for maximum performance. In this case, only one variable is mapped to RAM because the default value for the new RAM Threshold project option is set to 64 (elements). Rather than apply a memmap directive to the other four variables listed below tap_delay, you can simply change the RAM Threshold project option to 48 as shown in the figure below.



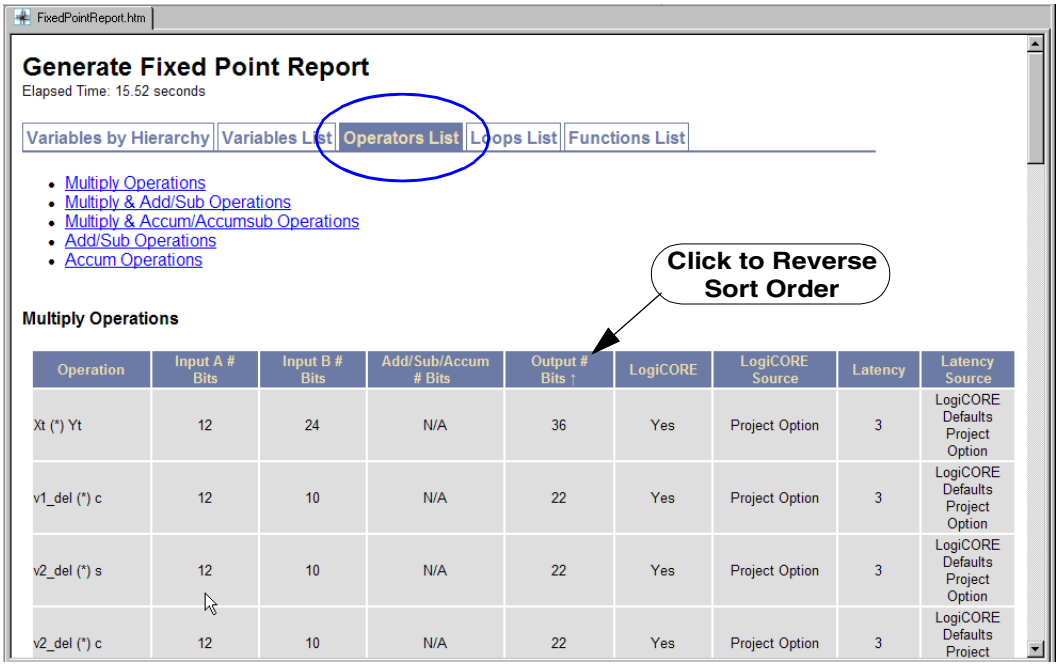
The following figure shows how four additional array variables are mapped to memory with one change in the RAM Threshold project option.



Variable	Shape	Elements ↑	Quantizer	Quantizer Source	Bits	Memory Map	Memory Map Source	Registered Output	Type
tap_delay	2 x 48	96	fixed floor wrap [16 0]	Auto Inferred	1536	dp_sync_ram	Automatically Inferred	Yes - Project Option	Double Persiste
DelayLine	3 x 20	60	fixed floor wrap [16 15]	MATLAB Source	960	dp_sync_ram	Automatically Inferred	Yes - Project Option	Double Persiste
DelayLine	3 x 20	60	fixed floor wrap [16 15]	MATLAB Source	960	dp_sync_ram	Automatically Inferred	Yes - Project Option	Double Persiste
CT	1 x 53	53	ufixed floor saturate [52 51]	**Hierarchical Directive**	2756	dp_sync_ram	Automatically Inferred	Yes - Project Option	Double Consta
coeff	48 x 1	48	fixed floor wrap [16 15]	MATLAB Source	768	dp_sync_ram	Automatically Inferred	Yes - Project Option	Double Persiste Consta

Operators List Report

The figure below shows the new Operators List within the Generate Fixed-Point Report.

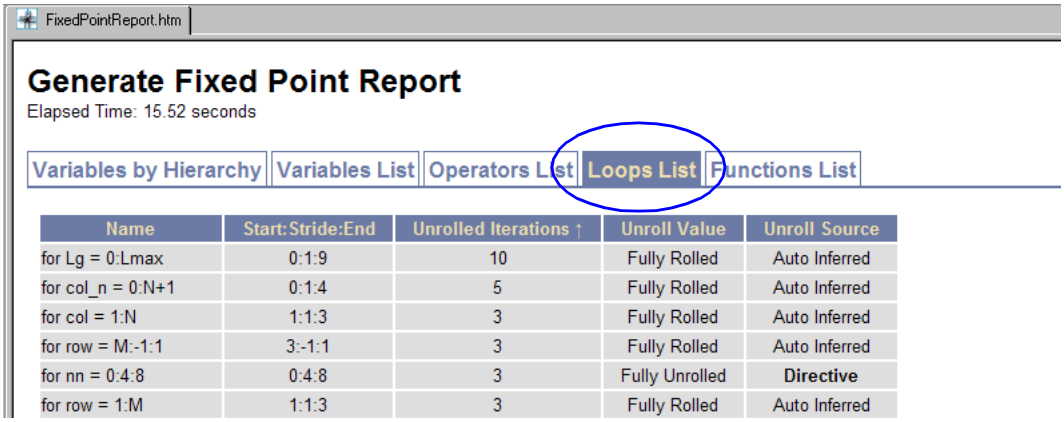


This report lists the name of each operator and provides important information about the operator parameters. You can change a parameter by clicking on the operator row, where you can change the parameter value in the Properties Viewer.

You can use this report to tune your design for maximum performance. For example, you can make sure that each operator is mapped to a LogiCORE, if possible, then change the LogiCORE latency to the desired value and adjust the quantization of each input. If you click on the column header of a particular parameter, the sort order for that column will reverse.

Loops List Report

The figure below shows the new Loops List within the Generate Fixed-Point Report.

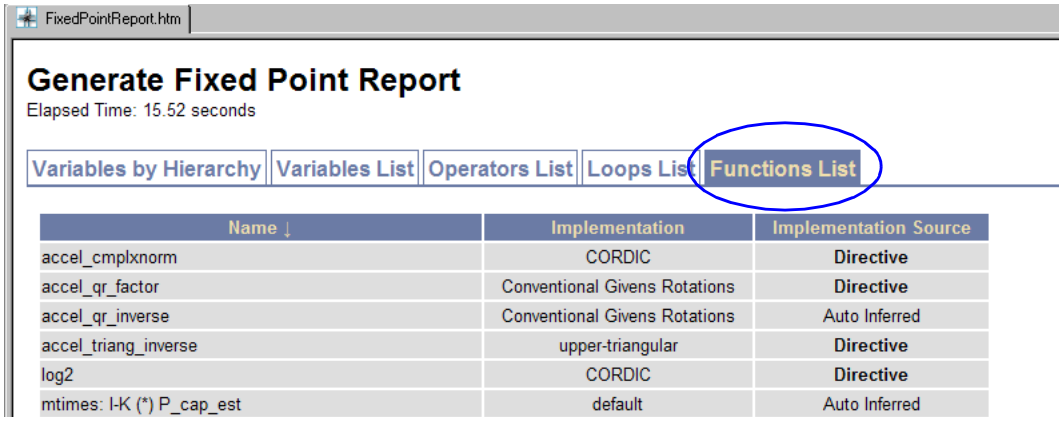


This report shows the start, stride and end values for each loop, the total number of unrolled iterations, the unroll values and the source of the unroll value. You can change the

unroll value of a particular loop by first clicking on the loop row, where you can change the unroll value in the Properties Viewer.

Functions List Report

The figure below shows the new Functions List within the Generate Fixed-Point Report.



This report provides the name of each inferred function, the implementation and the source of the implementation (Auto Inferred or Directive). You can change the implementation by clicking on the function row, where you can change the implementation value in the Properties Viewer.

AccelWare Tab and References Have Been Removed

The **Create AccelWare** tab in the Open Project dialog box and all references to the term "Accelware" have been removed from the product. All MATLAB functions that were previously supported with AccelWare are still fully supported by AccelDSP. Refer to the topic MATLAB for Synthesis Style Guide for detailed information on available functions.

System Requirements and Recommendations

Hardware Recommendations

Recommendation	Notes
2.00 GB of RAM	
600 MB of hard disk space	Minimum Requirement
Xilinx® Hardware Co-Simulation Platform	Required for the Hardware Co-Simulation Flow

Operating System and Software Requirements

Requirement	Notes
Windows XP 32 bit Operating System SP2	The AccelDSP synthesis tool does not support Windows Terminal Services.
Xilinx® ISE Design Suite Release 11.1	
MathWorks MATLAB® Version 2008a or 2008b	
MathWorks Simulink with Fixed-Point Toolbox Version 2008a or 2008b	Required for the System Generator Flow

Known Issues

Known issues with the AccelDSP synthesis tool can be found at the following Xilinx web site address:

http://www.xilinx.com/support/answers/29595.htm

XILINX®

Chapter 1

Release 10.1.3

AccelDSP Enhancements

New Directives

Use LogiCORE Directive

The use_logicore directive tells AccelDSP to use a LogiCORE for the specified operator(s) in the design. LogiCOREs supported in this release are Accumulator, Multiplier, Multiply Accumulator, and Multiply Adder. Typical MATLAB coding styles that support the inference of LogiCOREs are as follows:

Multiplier

```
a = b * c; % where b and c are non-constant scalar doubles.
```

Accumulator

```
acc = 0; % reset line
...
for i = 1:10
...
acc = acc + a(i); % accumulation line
...
end
```

Multiply Accumulator

```
acc = 0; % reset line
...
for i = 1:10
...
acc = acc + a(i) * b(i); % accumulation line
...
end
```

Multiply Adder

```
d = c + a * b;
```

Note: To infer a Multiply Adder, the multiply and add operators must be on the same MATLAB line.

You can also use this directive to override the Use LogiCOREs project option. For example, assume that the Use LogiCOREs project option is set to true and you discover that the automatic latency causes the performance to decrease because of design feedback. You can use this directive to disable the use of a LogiCORE for this one operator.

If an insertpipestage directive is also applied to the target operator, then AccelDSP will use the values of the insertpipestage directive to determine the latency of the LogiCORE. For example, consider the following combination of directives:

```
SetDirective {for n.arraymath.a (_*) d} -use_logicore 1
SetDirective {for n.arraymath.a (_*) d} -insertpipestage {before=1}
{after=2}
```

The use_logicore directive tells AccelDSP to use a LogiCORE for the multiply operator. Since the insertpipestage directive specifies a total of 3 registers, the latency of the mult LogiCORE to set to 3. If the number of registers specified by the insertpipestage directive is set to zero, then the latency of the LogiCORE is set to zero. If no insertpipestage directive is applied, then the latency for the LogiCORE is specified as "auto" and the tool determines the best latency for the design.

InsertPipeStage Directive - New enable Parameter

A new optional parameter call "enable" has been added to the insertpipestage directive. This allows you to specify whether or not an associated hierarchical directive is enabled or disabled.

Assume that your design function includes a sub-design that has a hierarchical directive applied to a multiplier. Assume also that you want AccelDSP to use a LogiCORE for the multiplier and that you want the LogiCORE to use "auto" for the latency. Since you cannot delete a hierarchical directive, you can effectively disable the hierarchical directive by applying a local insertpipestage directive with the "enable" parameter set to false(0). The directives might look similar to the following:

```
SetDirective {for n.hw.a*b (*) c.a (*) b} -use_logicore 1
SetDirective {for n.hw.a*b (*) c.a (*) b} -insertpipestage {enable=0}
```

Memmap Directive - New register_output Parameter

A new parameter called "register_output" has been added to the memmap directive. This allows you to specify whether or not the output of the memory is registered. For example:

```
SetDirective {for n.hw.a} -memmap {dp_sync_ram {register_output 1}}
SetDirective {for n.hw.b} -memmap {nomap {register_output 1}}
```

The first directive tells AccelDSP to map the array (named a) to BlockRAM, then add a register to the output. The second directive tells AccelDSP to map the array (named b) to registers in the fabric, then add a register to the output. These directives have a higher precedence over project option settings and global option settings.

Now assume that your design contains an array that is mapped to registers in the fabric and you set the project option memmap_defaults to true (1). AccelDSP will add an output register to this non-mapped array. Assume also that you get a warning message that says this array is in a feedback loop and that a simulation mismatch may occur. You can turn off the use of an output register on this particular array with a memmap directive similar to the following:

```
SetDirective {for n.arraymath.c} -memmap {nomap {register_output 0}}
```

Tools and Flow Integration

AccelDSP Synthesis is currently compatible with the following tools:

Table 1-1: **AccelDSP Compatibility with Other Tools**

Tool	Version
The MathWorks MATLAB®, Simulink	2007b and 2008a
Mentor Graphics ModelSim® SE	6.3c
Mentor Graphics Precision™ RTL Synthesis	2006a.101
Microsoft® Internet Explorer	6.0 or later
Synplicity Synplify Pro®	8.8.0.4 (requires a floating license from Synplicity for AccelDSP integrated optimization)
Xilinx® ISE	10.1.03
Xilinx® ISE Simulator	10.1.03
Xilinx® System Generator	10.1.03

Chapter 2

Release 10.1.2

AccelDSP Enhancements

New Global/Project Options

The following features have been added to AccelDSP and may be specified as Project Options or Global Options.

Memmap Defaults - Register_Output

When the register_output option is set to 1 (true), a register is inserted at the output of every memory read port in the design. In many cases, this will shorten the critical path and increase the maximum frequency. For example:

```
SetProjectOption -memmap_defaults {{register_output 1}}
```

This reduces the need to add separate insertpipestage directives after memory-mapped varibles to break the critical paths.

If a memory read operation happens to be in a feedback loop, a warning message will be issued indicating that a verification mismatch may result. In this case, it is best to turn this project option off (0).

Use LogiCORES

This project option turns on the use of Xilinx optimized LogiCORE IP in the generated HDL for all supported operators. The default is 0 (false). In this release the multiplier operator is supported. Additional operators will be supported in upcoming releases.

If you set this project option to 1 (true) and the overall performance of your design does not increase, then you should turn the project option back off (0).

MATLAB 2008a Support

MATLAB 2008a is now supported by AccelDSP.

Tools and Flow Integration

AccelDSP Synthesis is currently compatible with the following tools:

Table 2-1: **AccelDSP Compatibility with Other Tools**

Tool	Version
The MathWorks MATLAB®, Simulink	2007b and 2008a
Mentor Graphics ModelSim® SE	6.3c
Mentor Graphics Precision™ RTL Synthesis	2006a.101
Microsoft® Internet Explorer	6.0 or later
Synplicity Synplify Pro®	8.8.0.4 (requires a floating license from Synplicity for AccelDSP integrated optimization)
Xilinx® ISE	10.1.02
Xilinx® ISE Simulator	10.1.02
Xilinx® System Generator	10.1.02

XILINX®

Chapter 3

Release 10.1.1

Tools and Flow Integration

AccelDSP Synthesis is currently compatible with the following tools:

Table 3-1: **AccelDSP Compatibility with Other Tools**

Tool	Version
The MathWorks MATLAB®, Simulink Fixed-Point Toolbox	2007a and 2007b
Mentor Graphics ModelSim® SE	6.3c
Mentor Graphics Precision™ RTL Synthesis	2006a.101
Microsoft® Internet Explorer	6.0 or later
Synplicity Synplify Pro®	8.8.0.4 (requires a floating license from Synplicity for AccelDSP integrated optimization)
Xilinx® ISE	10.1.01
Xilinx® ISE Simulator	10.1.01
Xilinx® System Generator	10.1.01

Chapter 4

Release 10.1

AccelDSP Enhancements

Native Complex Number Support

AccelDSP can now synthesize MATLAB written using built-in complex numbers. For example, the following code can now be synthesized into RTL:

```
function y = my_design(x)
A = 3+4i;
y = (x + A) * -3i / 2;
```

Refer to the manual *MATLAB for Synthesis Style Guide* on supported functions.

More Efficient Mapping to BlockRAMs

Previously, AccelDSP could schedule a BlockRAM to be written to and read from in a single cycle. Now two values can be read from or written to a BlockRAM in a single cycle. This doubles the throughput attainable from each BlockRAM.

Tools and Flow Integration

AccelDSP Synthesis is currently compatible with the following tools:

Table 4-1: **AccelDSP Compatibility with Other Tools**

Tool	Version
The MathWorks MATLAB®, Simulink Fixed-Point Toolbox	2007a and 2007b
Mentor Graphics ModelSim® SE	6.3c
Mentor Graphics Precision™ RTL Synthesis	2006a.101
Microsoft® Internet Explorer	6.0 or later
Synplicity Synplify Pro®	8.8.0.4 (requires a floating license from Synplicity for AccelDSP integrated optimization)
Xilinx® ISE	10.1
Xilinx® ISE Simulator	10.1
Xilinx® System Generator	10.1