## Contents

1 About DSP Design............................................................................................................ 3  
   1.1 FPGA Architecture Features...................................................................................... 3  
   1.2 DSP Design Flow in FPGAs....................................................................................... 4  
   1.3 Software and Hardware Design Flows in FPGAs....................................................... 5  

2 About DSP Builder............................................................................................................ 6  
   2.1 About DSP Builder for Intel FPGAs ............................................................................ 6  
   2.2 About the Advanced and Standard Blocksets.......................................................... 6  
   2.3 Tool Integration...................................................................................................... 7  

3 Installing DSP Builder..................................................................................................... 8  
   3.1 System Requirements............................................................................................. 8  
   3.2 Installing DSP Builder for Intel FPGAs................................................................. 8  
   3.3 Licensing DSP Builder for Intel FPGAs............................................................... 10  

4 Document Revision History............................................................................................ 11
1 About DSP Design

1.1 FPGA Architecture Features

You can configure FPGAs to operate in different modes corresponding to a required functionality. You can use a suitable hardware description language (HDL) such as VHDL or Verilog HDL to implement any hardware design. Thus, the same FPGA can implement a DSL router, a DSL modem, a JPEG encoder, a digital broadcast system, or a backplane switch fabric interface.

High-density FPGAs incorporate embedded silicon features that can implement complete systems inside an FPGA, creating a system on a programmable chip (SOPC) implementation. Embedded silicon features such as embedded memory, DSP blocks, and embedded processors are ideally suited for implementing DSP functions such as finite impulse response (FIR) filters, fast Fourier transforms (FFTs), correlators, equalizers, encoders, and decoders.

The embedded DSP blocks also provide other functionality such as addition, subtraction, and multiplication, which are common arithmetic operations in DSP functions. Intel FPGAs offer much more multiplier bandwidth than DSP processors, which only offer a limited number of multipliers.

One determining factor of the overall DSP bandwidth is the multiplier bandwidth, therefore the overall DSP bandwidth of FPGAs can be much higher using FPGAs than with DSP processors.

Many DSP applications use external memory devices to manage large amounts of data processing. The embedded memory in FPGAs meets these requirements and also eliminates the need for external memory devices in some cases.

Embedded processors in FPGAs provide overall system integration and flexibility while partitioning the system between hardware and software. You can implement the system’s software components in the embedded processors and implement the hardware components in the FPGA’s general logic resources. Intel devices provide a choice between embedded soft core processors and embedded hard core processors.

You can implement soft core processors such as the Nios® II embedded processor in FPGAs and add multiple system peripherals. The Nios II processor supports a user-determinable multi-master bus architecture that optimizes the bus bandwidth and removes potential bottlenecks found in DSP processors. You can use multimaster buses to define as many buses and as much performance as needed for a particular application. Off-the-shelf DSP processors make compromises between size and performance when they choose the number of data buses on the chip, potentially limiting performance.
Soft embedded processors in FPGAs provide access to custom instructions such as the
MUL instruction in Nios II processors that can perform a multiplication operation in two
clock cycles using hardware multipliers. FPGA devices provide a flexible platform to
accelerate performance-critical functions in hardware because of the configurability of
the device’s logic resources. DSP processors have predefined hardware accelerator
blocks, but FPGAs can implement hardware accelerators for each application, allowing
the best achievable performance from hardware acceleration. You can implement
hardware accelerator blocks with parameterizable IP functions or from scratch using
HDL.

Intel offers many IP cores for DSP design. You can parameterize Intel DSP IP cores for
the most efficient hardware implementation and to provide maximum flexibility. You
can easily port the IP to new FPGA families, leading to higher performance and lower
cost. The flexibility of programmable logic and soft IP cores allows you to quickly
adapt your designs to new standards without waiting for long lead times usually
associated with DSP processors.

1.2 DSP Design Flow in FPGAs

Traditionally, system engineers use a hardware flow based on a HDL language, such as
Verilog HDL or VHDL, to implement DSP systems in FPGAs. Intel tools such as DSP
Builder, enable you to follow a software-based design flow while targeting FPGAs. DSP
Builder for Intel FPGAs simplifies hardware implementation of DSP functions, provides
a system-level verification tool to the system engineer who is not necessarily familiar
with HDL design flow, and allows the system engineer to implement DSP functions in
FPGAs without learning HDL. DSP Builder for Intel FPGAs provides an interface from
Simulink directly to the FPGA hardware. Additionally, you can incorporate the designs
created by DSP Builder for Intel FPGAs into a Qsys system for a complete DSP system
implementation.

**Figure 1. DSP Builder for Intel FPGAs General Design Flow for Intel FPGAs**

- Use MATLAB or Simulink to Design Algorithm
- Add Functions in DSP Builder
- Perform Synthesis, Place-and-Route (Quartus Prime Software)
- Evaluate Hardware in a DSP Development Kit
- DSP Libraries

---

**1 About DSP Design**

DSP Builder for Intel FPGAs Introduction Handbook

4
1.3 Software and Hardware Design Flows in FPGAs

Intel FPGAs with embedded processors support a software-based design flow. Intel provides the Nios II EDS development tools for compiling, debugging, assembling, and linking software designs. You can then use either on-chip RAM or an external memory device to download these software designs to an FPGA.

Embedded processors and hardware acceleration offer the flexibility, performance, and cost effectiveness in a development flow that is familiar to software developers. You can combine a software design flow with hardware acceleration. In this flow, you first profile C code and identify the functions that are the most performance critical. Then, you can use Intel's DSP IP or develop your own custom instructions to accelerate those tasks in the FPGA. You can run the system control code with the other low-performance DSP algorithms on a Nios II embedded processor. Intel also provides system integration tools such as Qsys for system-level partitioning and interconnection. You can use Qsys to build entire hardware systems by combining the embedded processor, such as a Nios II embedded processor, with other system peripherals and IP cores.

You can use an HDL-based hardware design flow to develop a pure hardware implementation of a DSP system. Intel provides a complete set of FPGA development tools including the Quartus® Prime software and interfaces to other EDA tools such as Synopsys, Synplify, and Precision Synthesis. These tools enable hardware design, simulation, debug, and in-system verification of the DSP system. You can also follow the DSP Builder for Intel FPGAs design flow and implement hardware-only DSP systems in FPGAs without learning HDL.
2 About DSP Builder

2.1 About DSP Builder for Intel FPGAs

DSP Builder for Intel FPGAs shortens DSP design cycles by helping you create the hardware representation of a DSP design in an algorithm-friendly development environment. DSP Builder for Intel FPGAs integrates the algorithm development, simulation, and verification capabilities of MathWorks MATLAB and Simulink system-level design tools with the Intel Quartus Prime software and third-party synthesis and simulation tools. You can combine Simulink blocks with DSP Builder for Intel FPGAs blocks to verify system level specifications and perform simulation.

Figure 3. DSP Builder for Intel FPGAs System-Level Design Flow

2.2 About the Advanced and Standard Blocksets

The DSP Builder for Intel FPGAs installer installs two separate blocksets (advanced and standard), which you can use separately or together from the Simulink library browser.

Note: The DSP Builder for Intel FPGAs standard blockset is a legacy product and Intel recommends you do not use it for new designs, except as a wrapper for advanced blockset designs.
2.3 Tool Integration

DSP Builder works with Simulink, the ModelSim simulator, and the Quartus Prime software (including Qsys).

Simulink

DSP Builder is interoperable with other Simulink blocksets. In particular, you can use the basic Simulink blockset to create interactive testbenches. The testbench block allows you to generate a VHDL model, so that you can compare Simulink simulation results with the ModelSim simulator. For information about Simulink fixed point types, the signal processing blockset and the communications blockset, refer to the MATLAB Help.

ModelSim Simulator

You can run the ModelSim simulator from within DSP Builder, if the ModelSim executable is in your path. You can use a script to integrate between the DSP Builder advanced blockset and the ModelSim simulator. The script runs the automatic testbench flow for a block. It reads some stimulus files at run time to verify a hardware block. The automatic testbench flow runs a rigorous test and returns a result whether or not the outputs match.

Quartus Prime Software

The advanced blockset allows you to build high-speed, high-performance DSP datapaths. In most production designs there is an RTL layer surrounding this datapath to perform interfacing to processors, high speed I/O, memories, and so on. To complete the design, use Qsys or RTL to assign board level components. The Quartus Prime software can then complete the synthesis and place-and-route process. You can automatically load a design into the Quartus Prime software by clicking on the Run Quartus Prime block in the top-level model.

Qsys

DSP Builder creates a conduit interface and hw.tcl file for each advanced blockset design. It creates a memory-mapped interface only if the design contains ModelBus blocks or external memory blocks. It can also create an Avalon® Streaming interface. The hw.tcl file can expose the processor bus for connection in Qsys. A DSP Builder advanced blockset subsystem is available from the System Contents tab in Qsys after you add the path to the hw.tcl file to the Qsys IP search path.
3 Installing DSP Builder

3.1 System Requirements

DSP Builder for Intel FPGAs integrates with the The MathWorks MATLAB and Simulink tools and with the Intel Quartus Prime software.

Ensure at least one version of The MathWorks MATLAB and Simulink tool is available on your workstation before you install DSP Builder for Intel FPGAs. You should use the same version of the Quartus Prime software and DSP Builder for Intel FPGAs. DSP Builder for Intel FPGAs only supports 64-bit versions of MATLAB.

Table 1. DSP Builder for Intel FPGAs Tool Dependencies

<table>
<thead>
<tr>
<th>Tool</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>DSP Builder for Intel FPGAs</td>
<td>17.0</td>
</tr>
<tr>
<td>The MathWorks (MATLAB and Simulink)</td>
<td>16.1</td>
</tr>
<tr>
<td>R2016a</td>
<td>16.0</td>
</tr>
<tr>
<td>R2015b</td>
<td></td>
</tr>
<tr>
<td>R2015a</td>
<td></td>
</tr>
<tr>
<td>R2014b</td>
<td></td>
</tr>
<tr>
<td>R2014a</td>
<td></td>
</tr>
<tr>
<td>R2013b</td>
<td></td>
</tr>
</tbody>
</table>

Note: The DSP Builder for Intel FPGAs advanced blockset uses Simulink fixed-point types for all operations and requires licensed versions of Simulink Fixed Point. Intel also recommends DSP System Toolbox and the Communications System Toolbox, which some design examples use.

Related Links

Intel Software Installation and Licensing.

3.2 Installing DSP Builder for Intel FPGAs

1. Install DSP Builder from the Intel Complete Design Suite.
   In the Intel software installer, ensure you turn on DSP Builder for Intel FPGAs in the Select components window.
The default installation directory is \altera\<version>\quartus on Windows or /opt/altera\<version>/quartus on Linux.

**Figure 5. DSP Builder for Intel FPGAs Directory Structure**

where <path> is the installation directory that contains the Quartus Prime software:

- installation directory containing the Quartus Prime software.
- dsp_builder
  - contains the DSP Builder standard blockset (legacy).
- dspba
  - contains the DSP Builder advanced blockset.
- backend
  - contains back-end files for various IP cores.
- blocksets
  - contains binary files and MATLAB scripts.
- devices
  - contains the device specifications.
- docs
  - contains the Simulink integrated help files.
- dspba_cockpit
  - contains GUI support files.
- examples
  - contains the design examples.
- libraries
  - contains extra HDL libraries.
- messages
  - contains error messages.
- polycache
  - contains floating-point support files.
- SysConAPI
  - contains the API files.
After installing DSP Builder for Intel FPGAs, the Intel DSP Builder standard blockset and the Intel DSP Builder advanced blockset libraries are available in the Simulink library browser in the MATLAB software.

3.3 Licensing DSP Builder for Intel FPGAs

Before you can use DSP Builder for Intel FPGAs, you must request a license file from the Altera website at and install it on your computer. The Quartus Prime software recommends you specify a path to an \texttt{LM_LICENSE\_FILE} variable, but it also allows you to use an explicit path to a license file. However, DSP Builder for Intel FPGAs allows you to specify a path to only an \texttt{LM_LICENSE\_FILE} variable.

\textbf{Related Links}

Intel Software Installation and Licensing.
# Document Revision History

DSP Builder introduction handbook revision history.

## Table 2. Document Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Software Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017.05.02</td>
<td>17.0</td>
<td>Updated MATLAB version support for DSP Builder v17.0.</td>
</tr>
<tr>
<td>2015.05.01</td>
<td>15.0</td>
<td>Updated MATLAB version support for DSP Builder v15.0.</td>
</tr>
<tr>
<td>June 2014</td>
<td>14.0</td>
<td>Updated MATLAB version support for DSP Builder v14.0.</td>
</tr>
<tr>
<td>May 2013</td>
<td>13.0</td>
<td>Updated MATLAB version support for DSP Builder v13.0.</td>
</tr>
<tr>
<td>November 2012</td>
<td>12.1</td>
<td>Updated MATLAB version support.</td>
</tr>
<tr>
<td>June 2012</td>
<td>12.0</td>
<td>• Updated MATLAB version support</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Deleted</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Upgrading from v7.1 chapter</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated installation instructions</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated instructions for starting DSP Builder</td>
</tr>
<tr>
<td>November 2011</td>
<td>11.1</td>
<td>Updated MATLAB version support.</td>
</tr>
<tr>
<td>April 2011</td>
<td>11.0</td>
<td>• Updated MATLAB version support</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Added support for 64-bit MATLAB</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Updated installation instructions</td>
</tr>
</tbody>
</table>