1. Starting DSP Builder in MATLAB on page 1
2. Browsing DSP Builder Libraries and Adding Blocks to a New Model on page 1
3. Browsing and Opening DSP Builder Design Examples on page 2
4. Creating a New DSP Builder Design with the DSP Builder New Model Wizard on page 3
   - Altera recommends you create new designs with the DSP Builder New Model Wizard. Alternatively, you can copy and rename a design example.
5. Simulating, Verifying, Generating, and Compiling Your DSP Builder Design on page 6
6. Primitive Library Blocks Tutorial on page 7
   - This tutorial shows how to build a simple design example that uses blocks from the Primitive library to generate a Fibonacci sequence.
7. Document Revision History on page 11
   - Changes for the DSP Builder Advanced Blockset Getting Started User Guide.

Related Information
- IP Tutorial
- Primitives Tutorial

Starting DSP Builder in MATLAB

1. On Windows OS, click Start > All Programs > Altera version > DSP Builder > Start in MATLAB version
2. Click the Simulink Library icon.

Related Information
- The DSP Builder Windows Shortcut Menu
  - Create the shortcut to set the file paths to DSP Builder and run a batch file with an argument for the MATLAB executable to use.
- Browsing DSP Builder Libraries and Adding Blocks to a New Model
- Browsing and Opening DSP Builder Design Examples

Browsing DSP Builder Libraries and Adding Blocks to a New Model

Before you begin
Start DSP Builder in MATLAB.
1. In MATLAB on the **Home** tab, click on the **Simulink Library** icon, to start Simulink.
2. In the Simulink Library Browser, in the left-hand pane, expand **DSP Builder Advanced Blockset**. Simulink lists the DSP Builder advanced blockset libraries.
3. Click on a library. Simulink shows the library in the right-hand pane.
4. To find more information about a block, right click on a block and click **Help for the block**.
5. To add a block to a model, right click on a block and click **Add block to a new model**.

**Related Information**
- **Starting DSP Builder in MATLAB**
- **Browsing and Opening DSP Builder Design Examples**
- **DSP Builder Advanced Blockset Libraries**
- **Creating a DSP Builder Design in Simulink**
  Altera recommends you create new designs with the DSP Builder New Model Wizard or copy and rename a design example.

### Browsing and Opening DSP Builder Design Examples

**Before you begin**

Start DSP Builder in MATLAB.
1. In MATLAB, on the **Home** tab, click the **Help** icon. The **Help** window opens.
2. Under **Supplemental software**, click **Altera DSP Builder Advanced Blockset**.
3. In the left-hand TOC pane, expand **DSP Builder Advanced Blockset Design Examples and Reference Designs**.
4. Expand **DSP Builder Advanced Blockset**.
   For example, to see the floating-point design examples, expand **Floating Point**.
5. Click on a design example to see a description.
6. Click **Open this model**, to open the design example.
7. You can also open a design example by typing a command in the MATLAB window, for example:

   ```matlab
demo_nco
```

**Related Information**
- **Starting DSP Builder in MATLAB**
- **DSP Builder Advanced Blockset Libraries**
- **Browsing DSP Builder Libraries and Adding Blocks to a New Model**
- **Creating a DSP Builder Design in Simulink**
  Altera recommends you create new designs with the DSP Builder New Model Wizard or copy and rename a design example.
Creating a New DSP Builder Design with the DSP Builder New Model Wizard

Altera recommends you create new designs with the DSP Builder New Model Wizard. Alternatively, you can copy and rename a design example.

Before you begin

Start DSP Builder in MATLAB.

1. In the Simulink Library browser, click New Model.
2. Click DSP Builder > New Model Wizard.
   The New Model Wizard opens.
3. Select a fixed- or floating-point model.
4. Select the type (simple or with channelizer).
5. Enter the model name and select where to save the model.

DSP Builder creates a new model <model name>.mdl and setup script setup_<model name>.m that contains everything you need for a DSP Builder model. DSP Builder automatically runs the set-up script when you open the model and before each simulation. To open and edit the script, click on the Edit Params block in the model.

Note: When you open a model, DSP Builder produces a model_name_params.xml file that contains settings for the model. You must keep this file with the model.

DSP Builder Menu Options on page 3
Simulink includes a DSP Builder menu on any Simulink model window. Use this menu to start all the common tasks you need to perform on your DSP Builder model.

DSP Builder New Model Wizard Setup Script Parameters on page 5
The setup script sets name-spaced workspace variables that DSP Builder uses to configure the design

Related Information

- Starting DSP Builder in MATLAB
- DSP Builder Advanced Blockset Libraries
- Simulating, Generating, and Compiling Your Design
- DSP Builder Menu Options
  Simulink includes a DSP Builder menu on any Simulink model window. Use this menu to easily start all the common tasks you need to perform on your DSP Builder model.
- DSP Builder New Model Wizard Setup Script Parameters
  Use the setup script to set name-spaced workspace variables that DSP Builder uses to configure the design
- DSP Builder Design Rules and Recommendations
  Obey the design rules and recommendations to ensure your design performs faultlessly

DSP Builder Menu Options

Simulink includes a DSP Builder menu on any Simulink model window. Use this menu to start all the common tasks you need to perform on your DSP Builder model.
### Table 1: DSP Builder Menu Options

<table>
<thead>
<tr>
<th>Action</th>
<th>Menu Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create new design</td>
<td>New Model Wizard</td>
<td>Create a new model from a simple template.</td>
</tr>
<tr>
<td></td>
<td>New SIL Wizard</td>
<td>Create a version of the existing design setup for hardware cosimulation.</td>
</tr>
<tr>
<td>Verification</td>
<td>Design Checker</td>
<td>Verify your design against basic design rules.</td>
</tr>
<tr>
<td></td>
<td>Verify Design</td>
<td>Verify the Simulink simulation matches ModelSim simulations of the generated hardware by batch running the automatically generated testbenches.</td>
</tr>
<tr>
<td>Parameterization</td>
<td>Avalon Interfaces …</td>
<td>Configure the memory mapped interface.</td>
</tr>
<tr>
<td>Generated hardware details</td>
<td>Resource Usage …</td>
<td>View resource estimates of the generated hardware.</td>
</tr>
<tr>
<td></td>
<td>Memory Map…</td>
<td>View the generated memory map interface.</td>
</tr>
<tr>
<td>Run other software tools</td>
<td>Run Quartus Prime Software</td>
<td>Run a Quartus Prime project for the generated hardware.</td>
</tr>
<tr>
<td></td>
<td>Run ModelSim</td>
<td>Verify the Simulink simulation matches ModelSim simulation of the generated hardware by running an automatically generated testbench in an open ModelSim window.</td>
</tr>
</tbody>
</table>
DSP Builder New Model Wizard Setup Script Parameters

The setup script sets name-spaced workspace variables that DSP Builder uses to configure the design.

The setup script offers the following options:

- Fixed-point IP (simple testbench)
- Fixed-point IP (with Channelizer)
- Fixed-point Primitive subsystem (simple testbench)
- Fixed-point Primitive subsystem (with Channelizer)
- Floating-point Primitive subsystem (simple testbench)
- Floating-point Primitive subsystem (with Channelizer)

Table 2: Setup Script Parameters

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Floating</td>
<td>The testbench propagates single precision floating-point data into the synthesizable system.</td>
</tr>
<tr>
<td>Fixed</td>
<td>The testbench propagates signed fixed-point data into the synthesizable system.</td>
</tr>
<tr>
<td>' (simple testbench)'</td>
<td>The testbench consists of simple Simulink source blocks.</td>
</tr>
<tr>
<td>Channelizer</td>
<td>The testbench consists of a Channelizer block, which outputs data from a MATLAB array in the DSP Builder valid-channel-data protocol.</td>
</tr>
<tr>
<td>'IP'</td>
<td>The synthesizable system has two IP function-level subsystems (IP library blocks) and FIR and a Scale block.</td>
</tr>
<tr>
<td>'Primitive'</td>
<td>The synthesizable system is a scheduled primitive subsystem with ChannelIn and ChannelOut boundary blocks. Use this start point to create your own function using low-level (primitive) building blocks.</td>
</tr>
</tbody>
</table>

Related Information

- **Creating a New DSP Builder Design with the DSP Builder New Model Wizard**
  Altera recommends you create new designs with the DSP Builder New Model Wizard. Alternatively, you can copy and rename a design example.

- **DSP Builder Menu Options**
  Simulink includes a DSP Builder menu on any Simulink model window. Use this menu to easily start all the common tasks you need to perform on your DSP Builder model.
Simulating, Verifying, Generating, and Compiling Your DSP Builder Design

Before you begin

- Create a design
- Check your design for errors

1. In Simulink, click **Simulation** > **Run**.
   
   **Note:** Simulink generates the HDL then starts the simulation

2. Analyze the simulation results.

3. Verify generated hardware (optional).
   a. Click **DSP Builder Verify Design**.
   b. Turn on **Verify at subsystem level**, turn off **Run Quartus Prime Software**, and click **Run Verification**.
      
      **Note:** If you turn on **Run Quartus Prime Software**, the verification script also compiles the design in the Quartus Prime software. MATLAB reports the postcompilation resource usage details in the verification window.

      MATLAB verifies that the Simulink simulation results match a simulation of the generated HDL in the ModelSim simulator.

   c. Close both verification windows when MATLAB completes the verification.

4. Examine the generated resource summaries:
   a. Click **Simulation** > **Start**.
   b. Click **Resource Usage** > **Design** for a top-level design summary.

5. View the Avalon-MM register memory map:
   a. Click **Simulation** > **Start**.
   b. Click **Memory Map** > **Design**. DSP Builder highlights in red any memory conflicts.

   **Note:** DSP Builder also generates the memory map in the `<design name>_mmap.h` file.

6. Compile your design in the Quartus Prime software by clicking **Run Quartus Prime**. When the Quartus Prime software opens, click **Processing** > **Start Compilation**.

Related Information

- **Creating a New DSP Builder Design with the DSP Builder New Model Wizard**
  Altera recommends you create new designs with the DSP Builder New Model Wizard. Alternatively, you can copy and rename a design example.

- **Creating a New Design by Copying a DSP Builder Design Example**

- **DSP Builder Advanced Blockset Generated Files**
  DSP Builder generates the files in a directory structure at the location you specify in the **Control** block, which defaults to `.rtl` (relative to the working directory that contains the `.mdl` file)

- **Control**
  The **Control** block specifies information about the hardware generation environment and the top-level memory-mapped bus interface widths.
Primitive Library Blocks Tutorial

This tutorial shows how to build a simple design example that uses blocks from the Primitive library to generate a Fibonacci sequence.

The Fibonacci sequence is the sequence of numbers that you can create when you add 1 to 0 then successively add the last two numbers to get the next number: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, ...

Each Primitive library block in the design example is parameterizable. When you double-click a block in the model, a dialog box appears where you can enter the parameters for the block. Click the Help button in these dialog boxes to view help for a specific block.

You can use the demo_fibonacci.mdl model in the <DSP Builder Advanced install path>/Examples/Primitive directory or you can create your own Fibonacci model.

Creating a Fibonnaci Design from the DSP Builder Primitive Library

Before you begin

Start DSP Builder in MATLAB

1. From an open Simulink model click DSP Builder > New Model Wizard.

   Note: When you open a model, DSP Builder produces a model_name_params.xml file that contains settings for the model. You must keep this file with the model.

2. Specify the following New Model Settings:

   • Fixed
   • Fixed-point Primitive (simple)
   • my_fibonacci

3. Browse to an appropriate output directory.

4. Click Generate.

5. In the Simulink Library Browser, click DSP Builder Advanced Blockset Primitive Primitive Basic Blocks.

6. Open the my_fibonacci generated model.

7. Open the dut > prim subsystem, which has a ChannelIn and ChannelOut block.

8. Drag and drop two SampleDelay blocks into your model.

   Note: Specify the data type because this design contains loops and DSP Builder cannot determine the type if one of the inherit data options is set.
9. Select both of the SampleDelay blocks and point to Rotate and Flip on the popup menu and click Flip Block to reverse the direction of the blocks.

10. Drag and drop Add and Mux blocks into your model.

11. Drag and drop a Const block. Double-click the block and:
   a. Select Specify via Dialog for Output data type mode.
   b. For Output type enter ufix(1).
   c. For Scaling enter 1
   d. For Value enter 1.
   e. Click OK.

12. Connect the blocks.

13. Double-click on the second SampleDelay block (SampleDelay1) to display the Function Block Parameters dialog box and change the Number of delays parameter to 2.

14. Double-click on the Add block to display the Function Block Parameters dialog box and set the parameters:
   a. For Output data type mode, select Specify via Dialog.
   b. For Output type enter ufix(120).
   c. For Output scaling value enter 2^-0
   d. For Number of inputs enter 2.
   e. Click OK.

Related Information

- Starting DSP Builder in MATLAB on page 1
- Creating a New DSP Builder Design with the DSP Builder New Model Wizard on page 3
Setting the Parameters on the Testbench Source Blocks

Set the testbench parameters to finish the DSP Builder Fibonacci design.

1. Double-click on the Real block to display the Source Block Parameters dialog box.
2. Set the Vector of output values to `[0 1 1 ones(1,171)]`, in the Main tab.
3. Switch to the Editor window for `setup_my_fibonacci.m`.
4. Change the parameters to:
   - `my_fibonacci_param.ChanCount = 1;`
   - `my_fibonacci_param.SampleRate = fibonacci_param.ClockRate;`
   - `my_fibonacci_param.input_word_length = 1;`
   - `my_fibonacci_param.input_fraction_length = 0;`
5. In the top-level design, delete the ChannelView, the Scope Deserialized Outputs scope and any dangling connections.
6. Double-click the Convert block and make the input unsigned by changing:
   
   ```
   fixdt(1,fibonacci_param.input_word_length,fibonacci_param.input_fraction_length)
   to:
   fixdt(0,fibonacci_param.input_word_length,fibonacci_param.input_fraction_length)
   ```
7. Save the Fibonacci model.

Figure 3: Completed Fibonacci Model

Simulating the Fibonacci Design in Simulink

1. Click Simulation > Run.
2. Double-click on the Scope block and click Autoscale in the scope to display the simulation results.
Figure 4: Fibonacci Sequence in the Simulink Scope

Note: You can verify that the fib output continues to increment according to the Fibonacci sequence by simulating for longer time periods.

The sequence on the fib output starts at 0, and increments to 1 when q_v and q_c are both high at time 21.0. It then follows the expected Fibonacci sequence incrementing through 0, 1, 1, 2, 3, 5, 8, 13 and 21 to 34 at time 30.0.

Related Information
Simulating, Verifying, Generating, and Compiling Your DSP Builder Design on page 6

Simulating the RTL of the Fibonacci Design

1. To verify that DSP Builder gives the same results when you simulate the generated RTL, click on the Run ModelSim block.
Compile the design in the Quartus Prime software.

Related Information
Simulating, Verifying, Generating, and Compiling Your DSP Builder Design on page 6

Document Revision History
Changes for the DSP Builder Advanced Blockset Getting Started User Guide.

<table>
<thead>
<tr>
<th>Date</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2016.02.24</td>
<td>Initial release</td>
</tr>
</tbody>
</table>