Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of “Quality Parts,Customers Priority,Honest Operation,and Considerate Service”,our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!

Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832
Email & Skype: info@chipsmall.com Web: www.chipsmall.com
Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China

DSP Builder Handbook

Volume 2: DSP Builder Standard Blockset

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

HB_DSPB_STD-16.0

Document last updated for Altera Complete Design Suite version: 16.0
Document publication date: August 2016

Feedback

Contents

Section I. DSP Builder Standard Blockset User Guide

Chapter 1. About DSP Builder Standard Blockset
Device Family Support . . . 1–1
Features . . . 1–1
General Description . . . 1–2
  High-Speed DSP with FPGAs . . . 1–2
  Interoperability with the Advanced Blockset . . . 1–3

Chapter 2. Getting Started
Creating a Design in DSP Builder . . . 2–1
Design Flow . . . 2–2
Creating an Amplitude Modulation Design Example . . . 2–4
  Creating a New Model . . . 2–4
  Adding the Sine Wave Block . . . 2–4
  Adding the SinIn Block . . . 2–6
  Adding the Delay Block . . . 2–7
  Adding the SinDelay and SinIn2 Blocks . . . 2–8
  Adding the Mux Block . . . 2–9
  Adding the Random Bitstream Block . . . 2–10
  Adding the Noise Block . . . 2–10
  Adding the Bus Builder Block . . . 2–11
  Adding the GND Block . . . 2–11
  Adding the Product Block . . . 2–11
  Adding the StreamMod and StreamBit Blocks . . . 2–12
  Adding the Scope Block . . . 2–13
  Adding a Clock Block . . . 2–14
Simulating the Model in Simulink . . . 2–15
Compiling the Design . . . 2–17
Performing RTL Simulation . . . 2–18
Adding the Design to a Quartus Prime Project . . . 2–21
  Creating a Quartus Prime Project . . . 2–21
  Adding the DSP Builder Design to the Project . . . 2–22

Chapter 3. Design Rules and Procedures
DSP Builder Naming Conventions . . . 3–1
MATLAB Variables . . . 3–2
Fixed-Point Notation . . . 3–2
  Binary Point Location in Signed Binary Fractional Format . . . 3–3
Bit Width Design Rule . . . 3–4
  Data Width Propagation . . . 3–4
    Tapped Delay Line . . . 3–6
    Arithmetic Operation . . . 3–6
Frequency Design Rules . . . 3–8
  Single Clock Domain . . . 3–8
  Multiple Clock Domains . . . 3–9
  Using Clock and Clock_Derived Blocks . . . 3–10
  Clock Assignment . . . 3–11

Using the PLL Block . . . 3–14
Using Advanced PLL Features . . . 3–15
About Timing Semantics Between Simulink and HDL Simulation . . . 3–16
Simulink Simulation Model . . . 3–16
HDL Simulation Models . . . 3–16
Startup and Initial Conditions . . . 3–17
Initial Reset of HDL Import Blocks and MegaCore Functions in Simulink Simulations . . . 3–17
DSP Builder Global Reset Circuitry . . . 3–17
Reference Timing Diagram . . . 3–18
Signal Compiler and TestBench Blocks . . . 3–19
Design Flows for Synthesis, Compilation and Simulation . . . 3–19
Hierarchical Design . . . 3–20
Goto and From Block Support . . . 3–21
Black Box and HDL Import . . . 3–22
Using a MATLAB Array or .hex File to Initialize a Block . . . 3–23
Comparison Utility . . . 3–23
Adding Comments to Blocks . . . 3–24
Adding Quartus II Constraints . . . 3–24
Displaying Port Data Types . . . 3–25
Displaying the Pipeline Depth . . . 3–25
Updating HDL Import Blocks . . . 3–26
Analyzing the Hardware Resource Usage . . . 3–26
Loading Additional ModelSim Commands . . . 3–28
About Quartus II Assignments and Block Entity Names . . . 3–28

Chapter 4. Using MegaCore Functions
Installing MegaCore Functions . . . 4–1
Updating MegaCore Function Variation Blocks . . . 4–2
Design Flow Using MegaCore Functions . . . 4–2
Adding the MegaCore Function in the Simulink Model . . . 4–2
Parameterizing the MegaCore Function Variation . . . 4–3
Generating the MegaCore Function Variation . . . 4–3
Connecting the MegaCore Function Variation Block to the Design . . . 4–3
Simulating the MegaCore Function Variation in the Model . . . 4–3
MegaCore Functions Design Issues . . . 4–3
Simulink Files Associated with a MegaCore Function . . . 4–3
About Simulating MegaCore Functions That Have a Reset Port . . . 4–4
About Setting the Device Family for MegaCore Functions . . . 4–4

Chapter 5. Using HIL
HIL Design Flow . . . 5–1
HIL Requirements . . . 5–2
HIL Design Example . . . 5–2
Burst Mode . . . 5–6
Using Burst Mode . . . 5–6
Troubleshooting HIL Designs . . . 5–7
Fails to Load the Specified Quartus Prime Project . . . 5–7
No Inputs Found From the Quartus Prime Project . . . 5–7
No Outputs Found From the Quartus Prime Project . . . 5–7
HIL Design Stays in Reset During Simulation . . . 5–7
HIL Compilation Appears to Hang . . . 5–8
Scan JTAG Fails to Find Correct Cable or Device . . . 5–8

Chapter 6. SignalTap II Logic Analysis
SignalTap II Design Flow 6–1
SignalTap II Nodes 6–2
SignalTap II Trigger Conditions 6–2
SignalTap II Design Examples 6–3
Opening the Design Example 6–3
Adding the Configuration and Connector Blocks 6–4
Specifying the Nodes to Analyze 6–5
Turning On the SignalTap II Option in Signal Compiler 6–5
Specifying the Trigger Levels 6–6
Performing SignalTap II Analysis 6–6

Chapter 7. Using the Interfaces Library
Avalon-MM Interface 7–1
Avalon-MM Interface Blocks 7–1
Avalon-MM Slave Block 7–2
Avalon-MM Master Block 7–3
Wrapped Blocks 7–4
Looking at the Avalon-MM Write FIFO 7–4
Looking at the Avalon-MM Read FIFO Buffer 7–5
Avalon-MM Interface Blocks Design Example 7–6
Adding Avalon-MM Blocks to the Design Example 7–6
Verifying the Design 7–9
Running Signal Compiler 7–9
Instantiating the Design in Qsys 7–10
Compiling the Quartus Prime Project 7–12
Testing the DSP Builder Block from Software 7–12
Avalon-MM FIFO Design Example 7–14
Opening the Design Example 7–14
Compiling the Design 7–15
Instantiating the Design in Qsys 7–16
Avalon-ST Interface 7–17
Avalon-ST Packet Formats 7–19

Chapter 8. Using Black Boxes for HDL Subsystems
Black Box Interfaces 8–1
Implicit Black Box Interface 8–1
Explicit Black-Box Interface 8–1
HDL Import Design Example 8–1
Importing Existing HDL Files 8–2
Simulating the HDL Import Model using Simulink 8–4
Subsystem Builder Design Example 8–6
Creating a Black Box System 8–6
Building the Black-Box SubSystem Simulation Model 8–8
Simulating the Subsystem Builder Model 8–11
Adding VHDL Dependencies to the Quartus Prime Project and ModelSim 8–11
Simulate the Design in ModelSim 8–12

Chapter 9. Using Custom Library Blocks
Creating a Custom Library Block 9–1
Creating a Library Model File 9–2
Building the HDL Subsystem Functionality 9–2
Defining Parameters Using the Mask Editor 9–3

Linking the Mask Parameters to the Block Parameters 9–4
Making the Library Block Read Only 9–5
Adding the Library to the Simulink Library Browser 9–5
Synchronizing a Custom Library 9–6

Chapter 10. Adding a Board Library
Creating a New Board Description 10–1
Predefined Components 10–1
Component Types 10–1
Component Description File 10–2
Example Component Description File 10–3
Board Description File 10–3
Header Section 10–3
Board Description Section 10–3
Building the Board Library 10–6

Chapter 11. Using the State Machine Library
Using the State Machine Editor Block 11–2

Chapter 12. Managing Projects and Files
Integrating with Source Control Systems 12–1
HDL Import 12–2
MegaCore Functions 12–2
Memory Initialization Files 12–2
About Exporting HDL 12–3
Using Exported HDL 12–4
Moving Standard Blockset Files to a New Location 12–4
Multiple Models in a Top-Level Quartus Prime Project 12–5
The ipx File Contents 12–6
Integrating Multiple Designs into a Top-Level Quartus Prime Project 12–6

Chapter 13. Troubleshooting
Signal Compiler Cannot Checkout a Valid License 13–1
Verifying That Your DSP Builder Licensing Functions Properly 13–2
Verifying That the LM_LICENSE_FILE Variable Is Set Correctly 13–3
Verifying the Quartus Prime Path 13–3
If You Still Cannot Get a License 13–4
Loop Detected While Propagating Bit Widths 13–4
The MegaCore Functions Library Does Not Appear in Simulink 13–4
The Synthesis Flow Does Not Run Properly 13–5
Check the Software Paths 13–5
DSP Development Board Troubleshooting 13–5
SignalTap II Analysis Appears to Hang 13–5
Error if Output Block Connected to an Altera Synthesis Block 13–5
Warning if Input/Output Blocks Conflict with clock or aclr Ports 13–6
Wiring the Asynchronous Clear Signal 13–6
Error Issues when a Design Includes Pre-v7.1 Blocks 13–6
Creating an Input Terminator for Debugging a Design 13–6
A Specified Path Cannot be Found or a File Name is Too Long 13–7
Incorrect Interpretation of Number Format in Output from MegaCore Functions 13–7
Simulation Mismatch For FIR Compiler MegaCore Function 13–7
Simulation Mismatch After Changing Signals or Parameters 13–7
Unexpected Exception Error when Generating Blocks 13–7

VHDL Entity Names Change if a Model is Modified 13–8
Algebraic Loop Causes Simulation to Fail 13–8
Parameter Entry Problems in the DSP Block Dialog Box 13–9
DSP Builder System Not Detected in Qsys 13–9
MATLAB Runs Out of Java Virtual Machine Heap Memory 13–9
ModelSim Fails to Invoke From DSP Builder 13–10
Unexpected End of File Error When Comparing Simulation Results 13–10

Section II. DSP Builder Standard Blockset Libraries

Chapter 14. AltLab Library
HDL Import Parameters 14–5
HDL Import Example 14–6
HDL Import Supported Megafunctions 14–6

Chapter 15. Arithmetic Library
Parameters 15–39

Chapter 16. Boards Library
Board Configuration 16–1
Adding PLL Output Clocks 16–1

Chapter 17. Complex Type Library

Chapter 18. Gate & Control Library

Chapter 19. Interfaces Library
PFC Data Flow 19–13
Packet Format Description 19–13
Packet Mapping 19–15
Multi-Packet Mapping 19–16
Error Handling 19–16

Chapter 20. IO & Bus Library

Chapter 21. MegaCore Functions Library

Chapter 22. Rate Change Library

Chapter 23. Simulation Library

Chapter 24. State Machine Functions Library

Chapter 25. Storage Library

Chapter 26. Design Examples
Viewing the Design Examples 26–2
Displaying a Model 26–2

Additional Information
Document Revision History Info–1

How to Contact Altera Info–2
Typographic Conventions Info–2

Section I. DSP Builder Standard Blockset User Guide

1. About DSP Builder Standard Blockset

Note: DSP Builder standard blockset is a legacy product and Altera recommends you do not use it for new designs, except as a wrapper for advanced blockset designs.

Device Family Support

DSP Builder supports the following Altera® device families:

- Arria V
- Arria V GZ
- Arria 10
- Cyclone IV GX
- Cyclone IV E
- Cyclone V
- Stratix® IV
- Stratix V

Features

DSP Builder standard blockset supports the following features:

- Altera DSP MegaCore® functions in a DSP Builder design model.
- Fixed-point arithmetic and logical operators for use with the Simulink software.
- Avalon® Memory-Mapped (Avalon-MM) interfaces including user configurable blocks, which you can use to build custom logic that works with the Nios® II processor and other Qsys designs.
- Avalon Streaming (Avalon-ST) interfaces including an Packet Format Converter block and configurable Avalon-ST sink and Avalon-ST source blocks.
- VHDL testbench.
- Rapid prototyping using Altera development boards.
- The SignalTap® II logic analyzer—an embedded signal analyzer that probes signals from the Altera device on the DSP board and imports the data into the MATLAB workspace to ease visual analysis.
- HDL import of VHDL or Verilog HDL design entities and HDL defined in a Quartus Prime project file.
- Hardware-in-the loop (HIL) to enable FPGA hardware accelerated cosimulation with Simulink.
- Tabular and graphical state machine editing.

For information about new features and errata in this release, refer to the *DSP Builder Release Notes*.

General Description

Digital signal processing (DSP) system design in Altera FPGAs requires both high-level algorithm and hardware description language (HDL) development tools.

Altera's DSP Builder integrates these tools by combining the algorithm development, simulation, and verification capabilities of The MathWorks MATLAB and Simulink system-level design tools with VHDL and Verilog HDL design flows, including the Altera Quartus Prime software.

DSP Builder shortens DSP design cycles by helping you create the hardware representation of a DSP design in an algorithm-friendly development environment.

You can combine existing MATLAB functions and Simulink blocks with Altera DSP Builder blocks and Altera intellectual property (IP) MegaCore functions to link system-level design and implementation with DSP algorithm development. In this way, DSP Builder allows system, algorithm, and hardware designers to share a common development platform.

The DSP Builder `Signal Compiler` block reads Simulink Model Files (**.mdl**) that contain other DSP Builder blocks and MegaCore functions. `Signal Compiler` then generates the VHDL files and Tcl scripts for synthesis, hardware implementation, and simulation.

You can use blocks from the standard blockset to create a hardware implementation of a system modeled in Simulink. DSP Builder contains bit- and cycle-accurate Simulink blocks—which cover basic operations such as arithmetic or storage functions—and takes advantage of key device features such as built-in PLLs, DSP blocks, and embedded memory.

You can integrate complex functions by including IP cores in your DSP Builder model. You can also use the faster performance and richer instrumentation of hardware cosimulation by implementing parts of your design in an FPGA.

The standard blockset supports imported HDL subsystems including HDL defined in a Quartus Prime project file.

High-Speed DSP with FPGAs

FPGAs give compelling performance advantage over dedicated DSP devices. You can configure FPGA's element arrays as complex processor routine.

You can link these routines together in serial (the same way that a DSP processor executes them), or connect them in parallel. When connected in parallel, they give many times better performance than standard DSP devices by executing hundreds of instructions at the same time.

Algorithms that benefit from this improved performance include forward-error correction (FEC), modulation and demodulation, and encryption.

Interoperability with the Advanced Blockset

Any standard blockset design can include an advanced blockset design as a single hierarchical entity.

For more information about the advanced blockset, refer to *Volume 3: DSP Builder Advanced Blockset* in the *DSP Builder Handbook*.

For more information about the differences between the standard and advanced blocksets and about design flows that combine both blocksets, refer to *Volume 1: Introduction to DSP Builder* in the *DSP Builder Handbook*.

2. Getting Started

This chapter describes the design flow and a getting started tutorial.

Creating a Design in DSP Builder

1. Create a Simulink design model in the MATLAB software.
2. Compile directly in the Quartus Prime software.
3. Output VHDL files for synthesis and Quartus II compilation or generate files for VHDL simulation.

☞ DSP Builder generates VHDL and does not generate Verilog HDL.

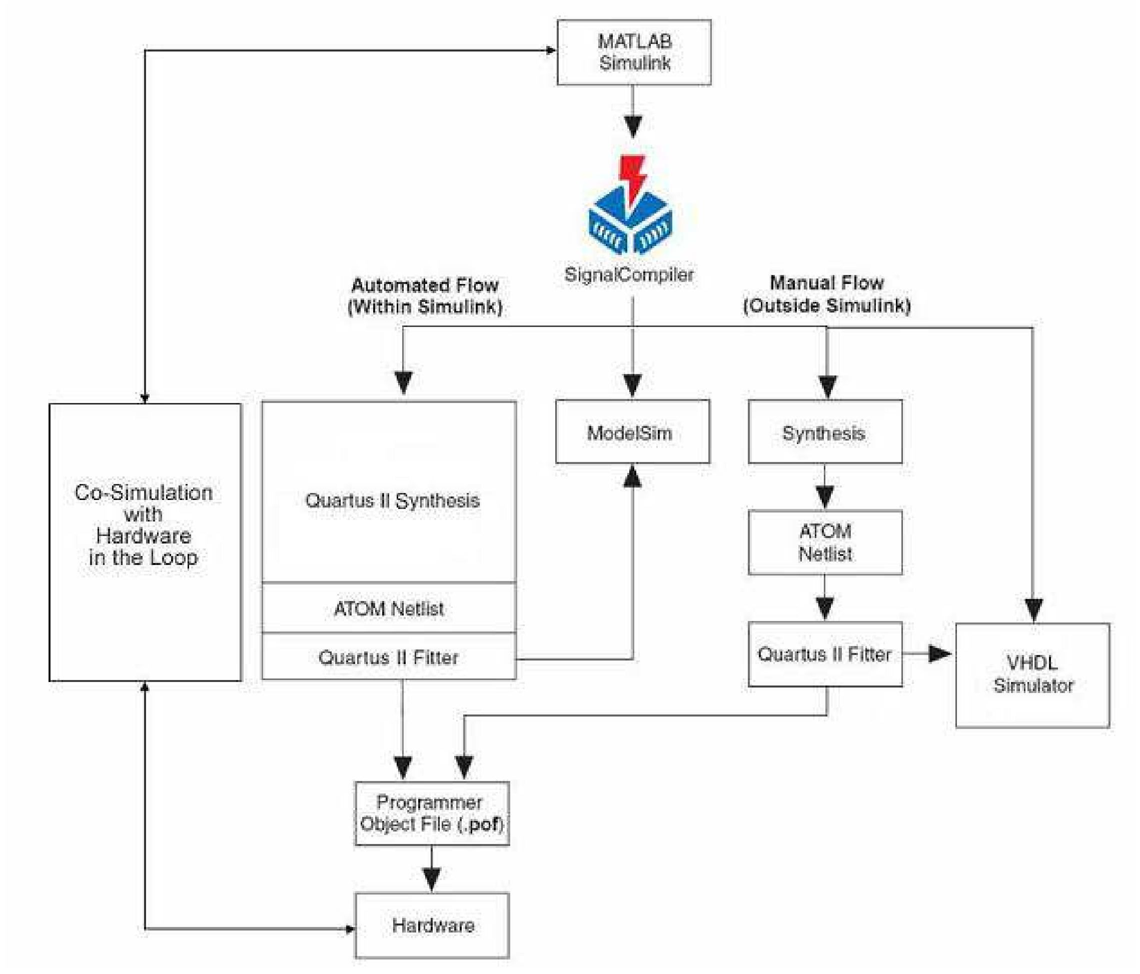
4. Use the `quartus_map` command in the Quartus Prime software to run a simulation netlist flow that generates files for Verilog HDL simulation.

For information about this flow, refer to the Quartus Prime help.

Design Flow

Figure 2–1 shows the system-level design flow using DSP Builder.

Figure 2–1. System-Level Design Flow



The design flow involves the following steps:

1. Use the MathWorks software to create a model with a combination of Simulink and DSP Builder blocks.

   ☞ Separate The DSP Builder blocks in your design from the Simulink blocks by `Input` and `Output` blocks from the DSP Builder IO and Bus library.

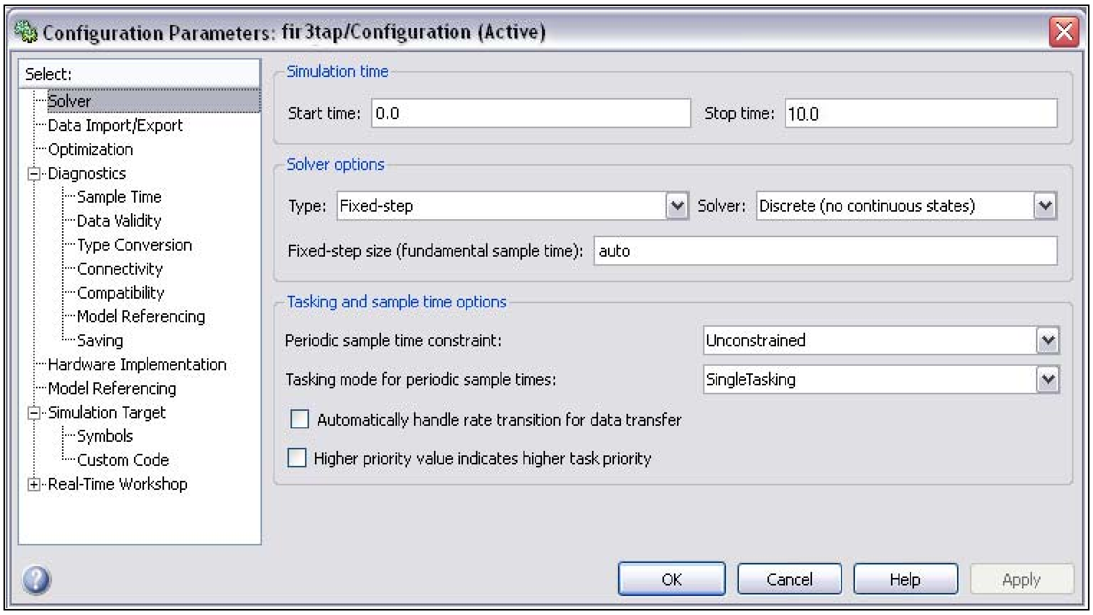
2. Include a `Clock` block from the DSP Builder AltLab library to specify the base clock for your design, which must have a period greater than 1ps but less than 2.1 ms.

   ☞ If no base clock exists in your design, DSP Builder creates a default clock with a 20ns real-world period and a Simulink sample time of 1. You can derive additional clocks from the base clock by adding `Clock_Derived` blocks.

3. Set a discrete (no continuous states) solver in Simulink. Choose a **Fixed-step** solver type if you are using a single clock domain or a **Variable-step** type if you use multiple clock domains.

   To set the solver options, click **Configuration Parameters** on the Simulation menu to open the **Configuration Parameters** dialog box and select the **Solver** page (Figure 2–2).

Figure 2–2. Configuration Parameters for Simulation



Start time: 0.0
Stop time: 10.0
Type: Fixed-step
Solver: Discrete (no continuous states)
Fixed-step size (fundamental sample time): auto
Periodic sample time constraint: Unconstrained
Tasking mode for periodic sample times: SingleTasking
☐ Automatically handle rate transition for data transfer
☐ Higher priority value indicates higher task priority

For detailed information about solver options, refer to the description of the “Solver Pane” in the Simulink Help.

4. Simulate your model in Simulink using a `Scope` block to monitor the results.
5. Run `Signal Compiler` to setup RTL simulation and synthesis.
6. Perform RTL simulation. DSP Builder supports an automated flow for the ModelSim software (using the `TestBench` block). You can also use the generated VHDL for manual simulation in other simulation tools.
7. Use the output files generated by the DSP Builder `Signal Compiler` block to perform RTL synthesis. Alternatively, you can synthesize the VHDL files manually using other synthesis tools.
8. Compile your design in the Quartus II software.
9. Download to a hardware development board and test.

For an automated design flow, the `Signal Compiler` block generates VHDL and Tcl scripts for synthesis in the Quartus II software. The Tcl scripts let you perform synthesis and compilation automatically in the MATLAB and Simulink environment. You can synthesize and simulate the output files in other software tools without the Tcl scripts. In addition, the `Testbench` block generates a testbench and supporting files for VHDL simulation.

For information about controlling the DSP Builder design flow using `Signal Compiler`, refer to "Design Flows for Synthesis, Compilation and Simulation" on page 3–19.

For more information about the blocks in the DSP Builder blockset, refer to the *DSP Builder Standard Blockset Libraries* section in volume 2 of the *DSP Builder Handbook*.

Creating an Amplitude Modulation Design Example

The amplitude modulation design example, **singen.mdl**, demonstrates the DSP Builder design flow.

The amplitude modulation design example is a modulator that has a sine wave generator, a quadrature multiplier, and a delay element. Each block in the model is parameterizable. When you double-click a block in the model, a dialog box displays where you can enter the parameters for the block. Click the **Help** button in these dialog boxes to view help for a specific block.

This tutorial assumes the following points:

- You are using a PC running Windows.
- You are familiar with the MATLAB, Simulink, Quartus II, and ModelSim® software and the software is installed on your PC in the default locations.
- You have basic knowledge of the Simulink software.

  For information about using the Simulink software, refer to the Simulink Help.

You can use the **singen.mdl** model file in <*DSP Builder install path*>**\DesignExamples\Tutorials\GettingStartedSinMdl** or you can create your own amplitude modulation model.

Creating a New Model

To create a new model:

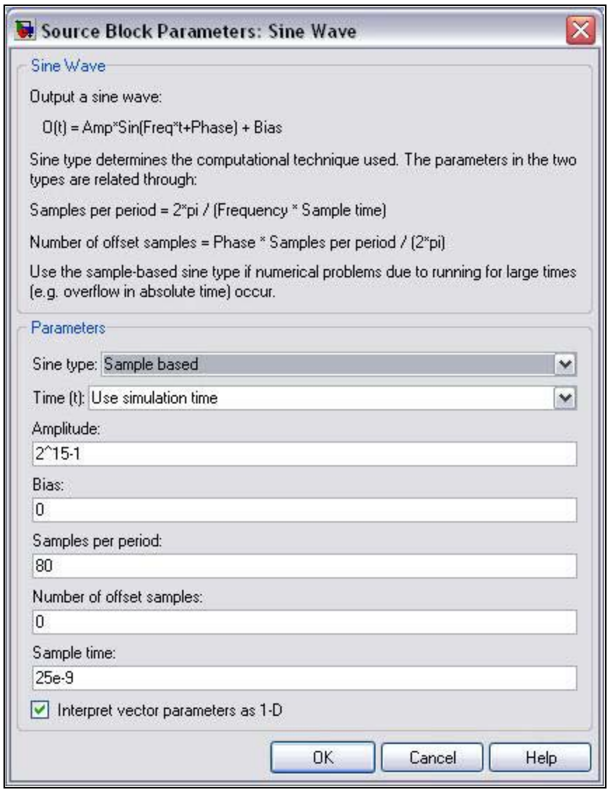
1. Start the MATLAB software.
2. On the File menu, point to **New,** and click **Model** to create a new model window.
3. In the new model window, on the File menu click **Save**.
4. Browse to a directory, your working directory, to save the file. This tutorial uses the working directory <*DSP Builder install path*>**\DesignExamples\Tutorials\GettingStartedSinMdl\my_SinMdl**.
5. Type the file name into the **File name** box. This tutorial uses the name **singen.mdl**.
6. Click **Save**.
7. Click the MATLAB **Start** button. Point to **Simulink** and click **Library Browser**.

Adding the Sine Wave Block

To add the `Sine Wave` block:

1. In the Simulink Library Browser, click **Simulink** and **Sources** to view the blocks in the Sources library.
2. Drag and drop a `Sine Wave` block into your model.
3. Double-click the `Sine Wave` block in your model to display the **Block Parameters** dialog box (Figure 2–3).

Figure 2–3. 500-kHz, 16-Bit Sine Wave Specified in the Sine Wave Dialog Box



Source Block Parameters: Sine Wave

Sine Wave

Output a sine wave:

O(t) = Amp*Sin(Freq*t+Phase) + Bias

Sine type determines the computational technique used. The parameters in the two types are related through:

Samples per period = 2*pi / (Frequency * Sample time)

Number of offset samples = Phase * Samples per period / (2*pi)

Use the sample-based sine type if numerical problems due to running for large times (e.g. overflow in absolute time) occur.

Parameters

Sine type: Sample based

Time (t): Use simulation time

Amplitude: 2^15-1

Bias: 0

Samples per period: 80

Number of offset samples: 0

Sample time: 25e-9

☑ Interpret vector parameters as 1-D

4. Set the `Sine Wave` block parameters (Table 2–1).

Table 2–1. Parameters for the Sine Wave Block

Parameter	Value
Sine type	Sample based
Time	simulation time
Amplitude	2^15–1
Bias	0
Samples per period	80
Number of offset examples	0
Sample time	25e-9
Interpret vector parameters a 1-D	On

5. Click **OK**.

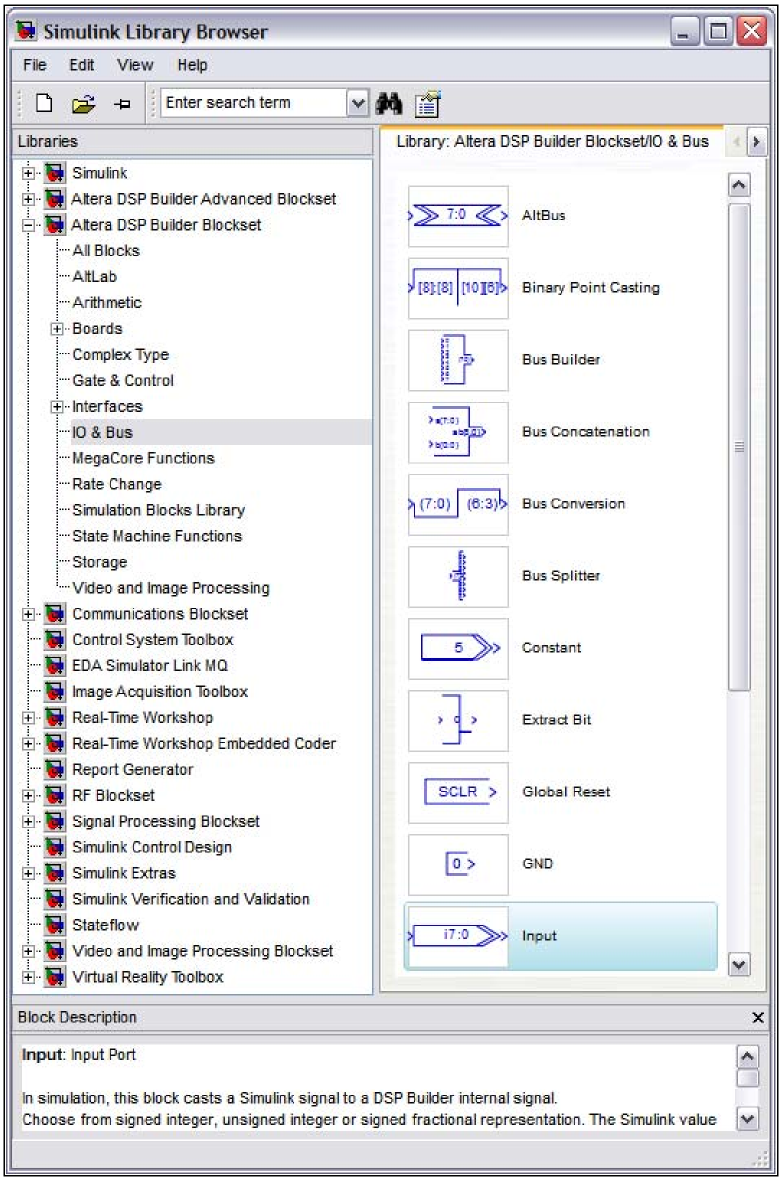
For information about how you can calculate the frequency., refer to the equation in "Frequency Design Rules" on page 3–8.

Adding the SinIn Block

To add the `SinIn` block:

1. In the Simulink Library Browser, expand the **Altera DSP Builder Blockset** folder to display the DSP Builder libraries (Figure 2–4).

Figure 2–4. Altera DSP Builder Folder in the Simulink Library Browser



Input: Input Port

In simulation, this block casts a Simulink signal to a DSP Builder internal signal.
Choose from signed integer, unsigned integer or signed fractional representation. The Simulink value

2. Select the **IO & Bus** library.

3. Drag and drop the `Input` block from the Simulink Library Browser into your model. Position the block to the right of the `Sine Wave` block.

   If you are unsure how to position the blocks or draw connection lines, refer to the completed design (Figure 2–7 on page 2–14).

   ☞ You can use the Up, Down, Right, and Left arrow keys to adjust the position of a block.

4. Click the text under the block icon in your model. Delete the text `Input` and type the text `SinIn` to change the name of the block instance.
5. Double-click the `SinIn` block in your model to display the **Block Parameters** dialog box.
6. Set the `SinIn` block parameters (Table 2–2).

Table 2–2. Parameters for the SinIn Block

Parameter	Value
Bus Type	Signed Integer
[number of bits].[]	16
Specify Clock	Off

7. Click **OK**.
8. Draw a connection line from the right side of the `Sine Wave` block to the left side of the `SinIn` block by holding down the left mouse button and dragging the cursor between the blocks.

   ☞ Alternatively, you can select a block, hold down the Ctrl key and click the destination block to automatically make a connection between the two blocks.

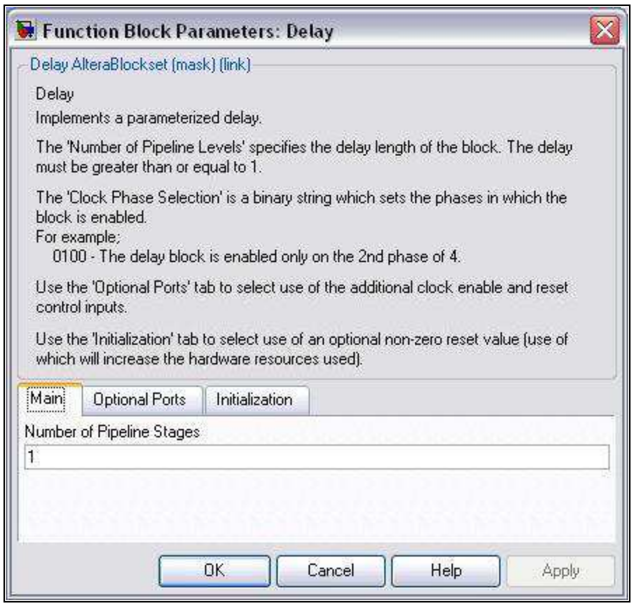
Adding the Delay Block

To add the `Delay` block:

1. Select the **Storage** library from the **Altera DSP Builder Blockset** folder in the Simulink Library Browser.
2. Drag and drop the `Delay` block into your model and position it to the right of the `SinIn` block.
3. Double-click the `Delay` block in your model to display the **Block Parameters** dialog box (Figure 2–5).

4. Type `1` as the **Number of Pipeline Stages** for the `Delay` block.

Figure 2–5. Setting the Downsampling Delay



Function Block Parameters: Delay

Delay AlteraBlockset (mask) (link)

Delay

Implements a parameterized delay.

The 'Number of Pipeline Levels' specifies the delay length of the block. The delay must be greater than or equal to 1.

The 'Clock Phase Selection' is a binary string which sets the phases in which the block is enabled.
For example;
0100 - The delay block is enabled only on the 2nd phase of 4.

Use the 'Optional Ports' tab to select use of the additional clock enable and reset control inputs.

Use the 'Initialization' tab to select use of an optional non-zero reset value (use of which will increase the hardware resources used).

Number of Pipeline Stages: 1

5. Click the **Optional Ports** tab and set the parameters (Table 2–3).

Table 2–3. Parameters for the Delay Block.

Parameter	Value
Clock Phase Selection	01
Use Enable Port	Off
Use Synchronous Clear port	Off

6. Click **OK**.
7. Draw a connection line from the right side of the `SinIn` block to the left side of the `Delay` block.

Adding the SinDelay and SinIn2 Blocks

To add the `SinDelay` and `SinIn2` blocks:

1. Select the **IO & Bus** library from the **Altera DSP Builder Blockset** folder in the Simulink Library Browser.
2. Drag and drop two `Output` blocks into your model, positioning them to the right of the `Delay` block.
3. Click the text under the block symbols in your model. Change the block instance names from `Output` and `Output1` to `SinDelay` and `SinIn2`.
4. Double-click the `SinDelay` block in your model to display the **Block Parameters** dialog box.

5. Set the `SinDelay` block parameters (Table 2–4).

Table 2–4. Parameters for the SinDelay Block

Parameter	Value
Bus Type	Signed Integer
[number of bits].[]	16
External Type	Inferred

6. Click **OK**.
7. Repeat steps 4 to 6 for the `SinIn2` block setting the parameters (Table 2–5).

Table 2–5. Parameters for the SinIn2 Block

Parameter	Value
Bus Type	Signed Integer
[number of bits].[]	16
External Type	Inferred

8. Draw a connection line from the right side of the `Delay` block to the left side of the `SinDelay` block.

Adding the Mux Block

To add the `Mux` block, follow these steps:

1. Select the Simulink **Signal Routing** library in the Simulink Library Browser.
2. Drag and drop a `Mux` block into your design, positioning it to the right of the `SinDelay` block.
3. Double-click the `Mux` block in your model to display the **Block Parameters** dialog box.
4. Set the `Mux` block parameters (Table 2–6).

Table 2–6. Parameters for the Mux Block

Parameter	Value
Number of Inputs	2
Display Options	bar

5. Click **OK**.
6. Draw a connection line from the bottom left of the `Mux` block to the right side of the `SinDelay` block.
7. Draw a connection line from the top left of the `Mux` block to the line between the `SinIn2` block.
8. Draw a connection line from the `SinIn2` block to the line between the `SinIn` and `Delay` blocks.

Adding the Random Bitstream Block

To add the `Random Bitstream` block, follow these steps:

1. Select the Simulink **Sources** library in the Simulink Library Browser.
2. Drag and drop a `Random Number` block into your model, positioning it underneath the `Sine Wave` block.
3. Double-click the `Random Number` block in your model to display the **Block Parameters** dialog box.
4. Set the `Random Number` block parameters (Table 2–7).

Table 2–7. Parameters for the Random number Block

Parameter	Value
Mean	0
Variance	1
Initial seed	0
Sample time	25e–9
Interpret vector parameters as 1-D	On

5. Click **OK**.
6. Rename the `Random Noise` block `Random Bitstream`.

Adding the Noise Block

To add the `Noise` block, follow these steps:

1. Select the **IO & Bus** library from the **Altera DSP Builder Blockset** folder in the Simulink Library Browser.
2. Drag and drop an `Input` block into your model, positioning it to the right of the `Random Bitstream` block.
3. Click the text under the block icon in your model. Rename the block `Noise`.
4. Double-click the `Noise` block to display the **Block Parameters** dialog box.
5. Set the `Noise` block parameters (Table 2–8).

Table 2–8. Parameters for the Noise Block

Parameter	Value
Bus Type	Single Bit
Specify Clock	Off

☞ The dialog box options change to display only the relevant options when you select a new bus type.

6. Click **OK**.
7. Draw a connection line from the right side of the `Random Bitstream` block to the left side of the `Noise` block.

Adding the Bus Builder Block

The `Bus Builder` block converts a bit to a signed bus. To add the `Bus Builder` block, follow these steps:

1. Select the **IO & Bus** library from the **Altera DSP Builder Blockset** folder in the Simulink Library Browser.
2. Drag and drop a `Bus Builder` block into your model, positioning it to the right of the `Noise` block.
3. Double-click the `Bus Builder` block in your model to display the **Block Parameters** dialog box.
4. Set the `Bus Builder` block parameters (Table 2–9).

Table 2–9. Parameters for the Bus Builder Block

Parameter	Value
Bus Type	Signer Integer
[number of bits].[]	2

5. Click **OK**.
6. Draw a connection line from the right side of the `Noise` block to the top left side of the `Bus Builder` block.

Adding the GND Block

To add the `GND` block, follow these steps:

1. Select the **IO & Bus** library from the **Altera DSP Builder Blockset** folder in the Simulink Library Browser.
2. Drag and drop a `GND` block into your model, positioning it underneath the `Noise` block.
3. Draw a connection line from the right side of the `GND` block to the bottom left side of the `Bus Builder` block.

Adding the Product Block

To add the `Product` block, follow these steps:

1. Select the **Arithmetic** library from the **Altera DSP Builder Blockset** folder in the Simulink Library Browser.
2. Drag and drop a `Product` block into your model, positioning it to the right of the `Bus Builder` block and slightly above it. Leave enough space so that you can draw a connection line under the `Product` block.
3. Double-click the `Product` block to display the **Block Parameters** dialog box.