1. Introduction to SOPC Builder

QII54001-9.1.0

Quick Start Guide

For a quick introduction on how to use SOPC Builder, follow these general steps:

- Install the Quartus® II software, which includes SOPC Builder. This is available at www.altera.com.
- Take advantage of the one-hour online course, *Using SOPC Builder*.
- Download and run the checksum sample design described in the *SOPC Builder Memory Subsystem Development Walkthrough* chapter in volume 4 of the *Quartus II Handbook*.

Overview

SOPC Builder is a powerful system development tool. SOPC Builder enables you to define and generate a complete system-on-a-programmable-chip (SOPC) in much less time than using traditional, manual integration methods. SOPC Builder is included as part of the Quartus II software.

You may have used SOPC Builder to create systems based on the Nios® II processor. However, SOPC Builder is more than a Nios II system builder; it is a general-purpose tool for creating systems that may or may not contain a processor and may include a soft processor other than the Nios II processor.

SOPC Builder automates the task of integrating hardware components. Using traditional design methods, you must manually write HDL modules to wire together the pieces of the system. Using SOPC Builder, you specify the system components in a GUI and SOPC Builder generates the interconnect logic automatically. SOPC Builder generates HDL files that define all components of the system, and a top-level HDL file that connects all the components together. SOPC Builder generates either Verilog HDL or VHDL equally.

In addition to its role as a system generation tool, SOPC Builder provides features to ease writing software and to accelerate system simulation. This chapter includes the following sections:

Architecture of SOPC Builder Systems

An SOPC Builder component is a design module that SOPC Builder recognizes and can automatically integrate into a system. You can also define and add custom components or select from a list of provided components. SOPC Builder connects multiple modules together to create a top-level HDL file called the SOPC Builder system. SOPC Builder generates system interconnect fabric that contains logic to manage the connectivity of all modules in the system.

SOPC Builder Modules

This document refers to *components* as the class definition for a module, for example a Nios® II processor. An *instance* is a parameterization of a component that's been added to a system, for example `cpu_0`.

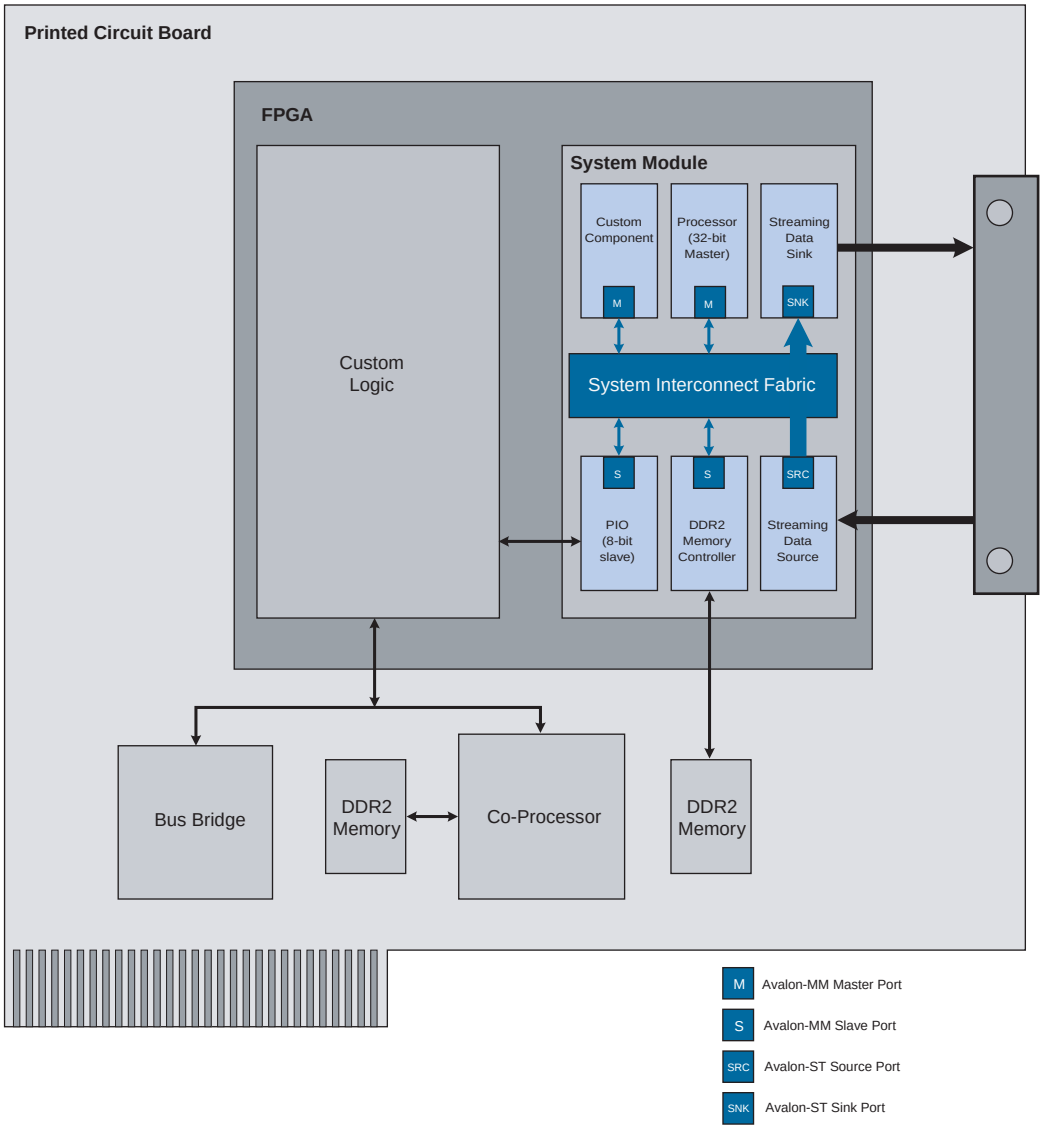
SOPC Builder modules are the building blocks for creating an SOPC Builder system. SOPC Builder modules use Avalon® interfaces, such as memory-mapped, streaming, and IRQ, for the physical connection of components. You can use SOPC Builder to connect any logical device (either on-chip or off-chip) that has an Avalon interface. There are different types of Avalon interfaces, as described in the *Avalon Interface Specifications*.

For details on the Avalon-MM interface refer to *System Interconnect Fabric for Memory-Mapped Interfaces* in chapter in volume 4 of the *Quartus II Handbook*. For details on the Avalon-ST interface, refer to the *System Interconnect Fabric for Streaming Interfaces* chapter in volume 4 of the *Quartus II Handbook*. For details about the Avalon-ST interface protocol, refer to *Avalon Interface Specifications*.

Example System

Figure 1–1 shows an FPGA design that includes an SOPC Builder system and custom logic modules. You can integrate custom logic inside or outside the SOPC Builder system. In this example, the custom component inside the SOPC Builder system communicates with other modules through an Avalon-MM master interface. The custom logic outside of the SOPC Builder system is connected to the SOPC Builder system through a PIO interface. The SOPC Builder system includes two SOPC Builder components with Avalon-ST source and sink interfaces. The system interconnect fabric connects all of the SOPC Builder components using the Avalon-MM or Avalon-ST system interconnect as appropriate.

Figure 1–1. Example of an FPGA with a SOPC Builder System Generated by SOPC Builder



A component can be a logical device that is entirely contained within the SOPC Builder system, such as the processor component shown in Figure 1–1. Alternately, a component can act as an interface to an off-chip device, such as the DDR2 interface component in Figure 1–1. In addition to the Avalon interface, a component can have other signals that connect to logic outside the SOPC Builder system. Non-Avalon signals can provide a special-purpose interface to the SOPC Builder system, such as the PIO in Figure 1–1. These non-Avalon signals are described in *Conduit Interface* chapter in the *Avalon Interface Specifications*.

Available Components

Altera and third-party developers provide ready-to-use SOPC Builder components, including:

- Microprocessors, such as the Nios II processor

- Microcontroller peripherals, such as a Scatter-Gather DMA Controller and timer
- Serial communication interfaces, such as a UART and a serial peripheral interface (SPI)
- General purpose I/O
- Communications peripherals, such as a 10/100/1000 Ethernet MAC
- Interfaces to off-chip devices

Custom Components

You can import HDL modules and entities that you write using Verilog HDL or VHDL into SOPC builder as custom components. You use the following design flow to integrate custom logic into an SOPC Builder system:

1. Determine the interfaces used to interact with your custom component.
2. Create the component logic using either Verilog HDL or VHDL.
3. Use the SOPC Builder component editor to create an SOPC Builder component with your HDL files.
4. Instantiate your component in the system.

Once you have created an SOPC Builder component, you can use the component in other SOPC Builder systems, and share the component with other design teams.

For instructions on developing a custom SOPC Builder component, the details about the file structure of a component, or the component editor, refer to the *SOPC Builder Components* chapter in volume 4 of the *Quartus II Handbook*.

For further details, refer to the *System Interconnect Fabric for Memory-Mapped Interfaces* and *System Interconnect Fabric for Streaming Interfaces* chapters in volume 4 of the *Quartus II Handbook*.

Third-Party Components

You can also use SOPC-ready components that were developed by third-parties. Altera awards the SOPC Builder Ready certification to IP functions that are ready to integrate with the Nios II embedded processor or the system interconnect fabric via SOPC Builder. These cores support the Avalon-MM interface or the Avalon Streaming (Avalon-ST) interface and may include constraints, software drivers, simulation models, and reference designs when applicable.

To find SOPC Builder Ready third-party components that you can purchase and use in SOPC Builder systems, complete the following steps:

1. On the Tools menu in SOPC Builder, click **Download Components**.
2. On the Intellectual Property Solutions web page, type `SOPC Builder ready` ↵ in the box labeled **Search for IP, Development Kits and Reference Designs**.

Functions of SOPC Builder

This section describes the functions of SOPC Builder.

Defining and Generating the System Hardware

SOPC Builder allows you to design the structure of a hardware system. The GUI allows you to add components to a system, configure the components, and specify connectivity.

After you add and parameterize components, SOPC Builder generates the system interconnect fabric, and outputs HDL files to your project directory. During system generation, SOPC Builder creates the following items:

- An HDL file for the top-level SOPC Builder system and for each component in the system. The top-level HDL file is named **<*system_name*>.v** for Verilog HDL designs and **<*system_name*>.vhd** for VHDL designs.
- Synopsis Design Constraints file (**.sdc**) for timing analysis.
- A Block Symbol File (.**bsf**) representation of the top-level SOPC Builder system for use in Quartus II Block Diagram Files (**.bdf**).
- An example of an instance of the top-level HDL file, **<*SOPC_project_name_inst*>.v** or **<*SOPC_project_name_inst*>.vhd**, which demonstrates how to instantiate the top-level HDL file in your code.
- A data sheet called <*system_name*>**.html** that provides a system overview including the following information:
  - All external connections for the system
  - A memory map showing the address of each Avalon-MM slave with respect to each Avalon-MM master to which it is connected
  - All parameter assignments for each component
- A functional test bench for the SOPC Builder system and ModelSim® simulation project files
- SOPC information file (**.sopcinfo)** that describes all of the components and connections in your system. This file is a complete system description, and is used by downstream tools such as the Nios II tool chain. It also describes the parameterization of each component in the system; consequently, you can parse its contents to get requirements when developing software drivers for SOPC Builder components.
- A Quartus II IP File (**.qip**) that provides the Quartus II software with all required information about your SOPC Builder system. The **.qip** file includes references to the following information:
  - HDL files used in the SOPC Builder system
  - TimeQuest Timing Analyzer Synopsys Design Constraint (**.sdc**) files
  - Component definition files for archiving purposes

After you generate the SOPC Builder system, you can compile it with the Quartus II software, or you can instantiate it in a larger FPGA design.

Creating a Memory Map for Software Development

When your SOPC Builder system includes a Nios II processor, SOPC Builder generates a header file, **cpu.h**, that provides the base address of each Avalon-MM slave component. In addition, each slave component can provide software drivers and other software functions and libraries for the processor. You can create C header files for your system using the `sopc-create-header-files` utility.

For details type `sopc-create-header-files --help` in a Nios II Command shell.

For more details about how to provide Nios II software drivers for components, refer to the *Developing Device Drivers for the Hardware Abstraction Layer* chapter of the *Nios II Software Developer's Handbook*. The Nios II EDS is separate from SOPC Builder, but it uses the output of SOPC Builder as the foundation for software development.

Creating a Simulation Model and Test Bench

You can simulate your system after generating it with SOPC Builder. During system generation, SOPC Builder outputs a simulation test bench and a ModelSim setup script that eases the system simulation effort. The test bench does the following:

- Instantiates the SOPC Builder system
- Drives all clocks and resets
- Instantiates simulation models for off-chip devices when available

Visualization of SOPC Builder Systems

You can use the **Filters** dialog box to customize the display of your system in the connections panel. You can filter the display of your system by interface type, instance name, interface type, or using custom tags. For example, you can use filtering to view only instances that include an Avalon-MM interface or instances that are connected to a particular Nios II processor. For more information, refer to Quartus II online Help.

Operating System Support

SOPC Builder supports all of the operating systems that the Quartus II software supports.

For details on installation and licensing, refer to the *Altera Software Installation and Licensing Manual*.

Talkback Support

Talkback is a Quartus II software feature that provides feedback to Altera on tool and IP feature usage. Altera uses the data to help guide future product planning efforts. Talkback sends Altera information on the Altera components you use, including: interface types, interface properties, parameter names and values, clocking, and software assignments. For components from Altera, Talkback sends the component parameter values to help understand what features of the component are being used. For non-Altera components, Talkback collects information about how interfaces such as Avalon-MM are being used. Connectivity between components is not sent. The Talkback file does not include information about system connectivity, interrupts, or the memory map seen by each master in the system. Talkback collects the same very general information about your proprietary components.

The Talkback feature is enabled by default. You can disable Talkback from within the Quartus II software if you do not wish to share your usage data with Altera.

Document Revision History

Table 1–1 shows the revision history for this chapter.

Table 1–1. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
November 2009, v9.1.0	■ Clarified information collected by Talkback.	—
March 2009, v9.0.0	■ Added sopc-create-header-files command ■ Added description of Generate HTML Data Sheet ■ Added instructions for downloading third-party IP. ■ Named top-level HDL system files that SOPC Builder generates. ■ Added paragraph introducing the filtering for visualization of large systems.	Updated to reflect new functionality in the 9.0 release.
November 2008, v8.1.0	■ Expanded description of **.sopcinfo** file ■ Changed page size to 8.5 x 11 inches	—
May 2008, v8.0.0	■ Updated references to Avalon Memory-Mapped and Streaming Interface Specifications and changed to Avalon Interface Specifications. ■ Add Quick Start Guide. ■ Add list of OS support.	The two specifications have been combined into one for all Avalon interfaces.
October 2007, v7.2.0	■ Updated with new 7.2 functionality and terminology. Deleted unneeded description of SOPC Builder Ready Components.	—

Table 1–1. Document Revision History

Date and Document Version	Changes Made	Summary of Changes
May 2007, v7.1.0	■ Updated Avalon terminology because of changes to Avalon technologies. Changed old "Avalon switch fabric" term to "system interconnect fabric." Changed old "Avalon interface" terms to "Avalon Memory-Mapped interface." ■ Added new information on Avalon Streaming (Avalon-ST) interface. ■ Revised SOPC Builder system block diagram ■ Added Referenced Documents section.	This chapter was revised to introduce the Avalon streaming interface in addition to the Avalon Memory-Mapped interface. The block diagram was made more comprehensive.
March 2007, v7.0.0	No change from previous release	—
November 2007, v6.1.0	No change from previous release.	—
May 2006, v6.0.0	No change from previous release.	—
October 2005, v5.1.0	No change from previous release.	—
May 2005, v5.0.0	No change from previous release.	—
February 2005, v1.0	Initial release.	—