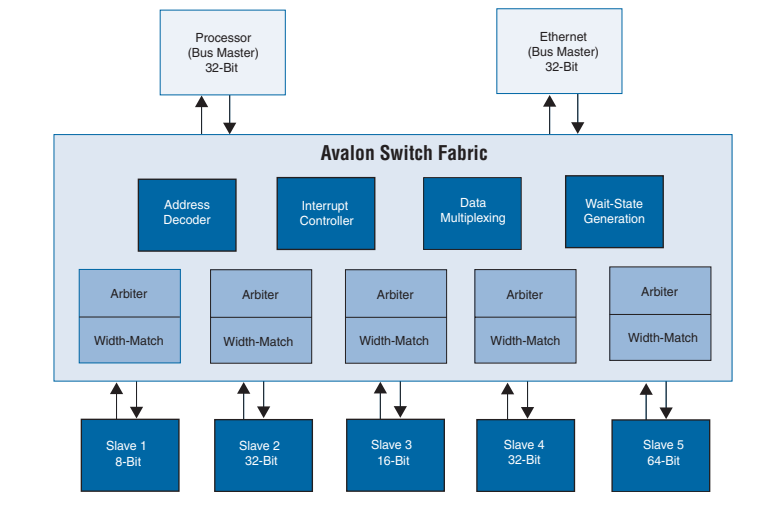


Introduction

The Altera® SOPC Builder system development tool provides a powerful platform for creating memory-mapped systems based on processors, peripherals, and memories that are internal or external to the FPGA. You can use SOPC Builder to define and implement a complete system in a fraction of the time required using traditional, manual system-on-a-chip (SoC) methods. SOPC Builder is included in Altera's Quartus® II software, giving Quartus II users immediate access to this development tool.

SOPC Builder automates the task of integrating the address-based read/write interfaces to hardware design modules. By integrating modules automatically, SOPC Builder dramatically simplifies the task of creating high-performance system-on-a-programmable-chip (SOPC) designs. In traditional SoC design, you must manually connect all of the system components. Using SOPC Builder, you need only specify the peripherals; SOPC Builder generates the interconnect logic automatically, including address decoding, data-path multiplexing, wait-state generation, interrupt controller, and data-width matching.

The outputs of SOPC Builder are hardware design language (HDL) files that define all components of the system, and a top-level HDL design file called the system module that ties all these components together. [Figure 2-1 on page 2-2](#) shows an example of a multi-master system module connecting multiple master and slave peripherals. SOPC Builder generates the Avalon™ switch fabric that contains logic to manage the connectivity of all modules in the system.

Figure 2–1. Example of a System Module Generated by SOPC Builder

SOPC Builder Peripherals

Altera and other developers provide SOPC Builder components that range from simple blocks of fixed logic, to complex, parameterized, and dynamically-generated subsystems. Available SOPC Builder hardware components include:

- Microprocessors
- Microcontroller peripherals
- Timers
- Serial communication interfaces, such as UART and serial peripheral interface (SPI)
- General-purpose I/O
- Digital signal processing (DSP) functions
- Communications peripherals
- Interfaces to off-chip devices
 - Memory controllers
 - Buses and bridges
 - Application-specific standard products (ASSPs)
 - ASICs

You can use the SOPC Builder to connect any block of logic that uses the Avalon interface or the AMBA™ advanced high-performance bus (AHB) interface. Most SOPC Builder peripherals use the Avalon interface.

Check the Altera web site at www.altera.com for up-to-date information about available SOPC Builder Ready components.

SOPC Builder Ready Functions

Altera awards the SOPC Builder Ready certification to intellectual property (IP) design functions that have plug-and-play integration with SOPC Builder. These functions may be accompanied by software drivers, low-level routines, or other software design files.

Altera offers a free “test drive” of IP functions through the OpenCore® and OpenCore Plus evaluation features. You can verify the functionality quickly and easily of a IP function both in simulation and in hardware, as well as evaluate the size and performance before making the purchase decision.



You can download OpenCore and OpenCore Plus evaluations of Altera IP functions directly from www.altera.com/IPMegastore. For IP functions provided by third-party vendors, contact the vendor directly to obtain an OpenCore evaluation.

User-Defined Peripherals

SOPC Builder provides a simple method for you to develop and connect your own modules:

1. You create a block of logic with an Avalon or AHB interface in either Verilog HDL or VHDL.

With the Avalon interface, user-defined peripherals need to adhere only to a simple interface based on address, data, read-enable, and write-enable signals.

2. Use the Interface to User Logic Wizard to import your HDL files into SOPC Builder.

You use the wizard to map the input and output signal names to Avalon signal types, specify the timing requirements, and specify simulation files.

3. Instantiate the custom module in the same manner as for other SOPC Builder Ready components.

User-defined peripherals can be instantiated multiple times in a single design, and can be used in other system designs.

Embedded Software Applications

You can specify and integrate software components for microprocessor-based systems. For each processor included in the system, SOPC Builder generates the following applications software-specific information:

- Software components
- Header files (C and assembly)
- Generic C drivers
- Operating system (OS) kernels
- Middleware libraries

These files form a custom software development kit (SDK) for each processor. For more details, see “[SDK Option](#)” on page 2–10.

Avalon Switch Fabric

The Avalon switch fabric is the glue that binds SOPC Builder-generated systems together. The Avalon switch fabric is the collection of control, data and address signals, and arbitration logic connecting master and slave peripherals. The Avalon switch fabric is implemented as a configurable architecture that matches the interface ports, logical behavior, and signal sequencing of the specific peripherals connected to the system.

The Avalon switch fabric is designed for both simplicity and performance. The Avalon signal types are straightforward, and you can design peripherals generically without knowing about the other peripherals connected to the system. The Avalon switch fabric implements a point-to-point connection between all master-slave pairs that require connection.

The multi-master Avalon switch fabric maximizes system throughput by allowing all master peripherals to transfer data in parallel. The Avalon switch fabric also supports pipelined transfers, so that master-slave pairs achieve the maximum throughput possible. SOPC Builder automatically implements slave-side arbitration whenever necessary. With slave-side arbitration, master transfers stall only when multiple master ports attempt to access the same slave port at the same time.

Automatic Generation

SOPC Builder generates the Avalon switch fabric to connect master and slave peripherals. Use the SOPC Builder to define the connectivity between peripherals. With this information, SOPC Builder automatically creates and connects the HDL modules.

Because of the SOPC Builder interface, your view of the Avalon switch fabric is usually limited to the specific ports that relate to the connection of custom Avalon peripherals. For SOPC Builder Ready IP functions, SOPC Builder automatically connects the Avalon ports correctly, making it unnecessary for you to consider the interfaces for these IP functions.

Function

The Avalon switch fabric provides the following services to connected peripherals:

- *Data-Path Multiplexing*—Multiplexers in the Avalon switch fabric transfer data from the selected slave peripheral to the appropriate master peripheral.
- *Address Decoding*—Address decoding logic produces chip-select signals for each peripheral. This simplifies peripheral design, because individual peripherals do not need to decode the address lines to generate chip-select signals.
- *Pipelined Transfer Capabilities*—Latency-aware peripherals are capable of initiating multiple read transfers in succession without waiting for the first transfer to complete, also known as pipelining data transfers. Transfers with latency allow master-slave pairs to achieve maximum throughput performance, even though the first transfer requires several cycles of latency to present valid data.
- *Wait-State Generation*—Wait-state generation extends transfers by one or more clock cycles, for the benefit of peripherals with special synchronization needs. Wait states can be generated to stall a master peripheral in cases when the target peripheral cannot respond in a single clock cycle. Wait states are also generated in cases when read-enable and write-enable signals have setup or hold time requirements.
- *Dynamic-Bus Sizing*—Dynamic-bus sizing hides the details of interfacing narrow Avalon slave ports to a wider Avalon master port, and vice versa. For example, in the case of a 32-bit master read transfer from a 16-bit memory, dynamic-bus sizing automatically executes two slave read transfers to fetch 32 bits of data from the 16-bit memory device. This reduces the logic and/or software complexity in master peripherals, because the master port does not need to be aware of the physical nature of the slave port.
- *Interrupt-Priority Assignment*—When one or more slave ports generate interrupts, the Avalon switch fabric passes the interrupts to appropriate master peripherals.

System Generation

After you add all peripherals and specify all necessary system parameters, SOPC Builder is ready to generate the system. During system generation, SOPC Builder creates the following items:

- HDL files for the top-level system module and each component in the system
- Block Symbol File (.bsf) representation of the top-level system module
- SDK for application software development
- ModelSim® simulation project files
- Tcl script that sets up all of the files needed for compilation in the Quartus II software

After you generate the system module, it can be compiled directly by the Quartus II software, or included in a larger design.

Simulation Model & Testbench

During system generation, SOPC Builder optionally can output a simulation environment that accelerates the system simulation effort. SOPC Builder generates both a simulation model and a testbench for the entire system. You can simulate your custom systems with minimal effort, immediately after generating the system with SOPC Builder. For more information, see the [“Simulation Option” on page 2–12](#).

Using SOPC Builder

Launch SOPC Builder in the Quartus II software by performing the following steps:

1. Open a project in the Quartus II software.
2. Choose **SOPC Builder** (Tools menu) in the Quartus II software to run SOPC Builder.

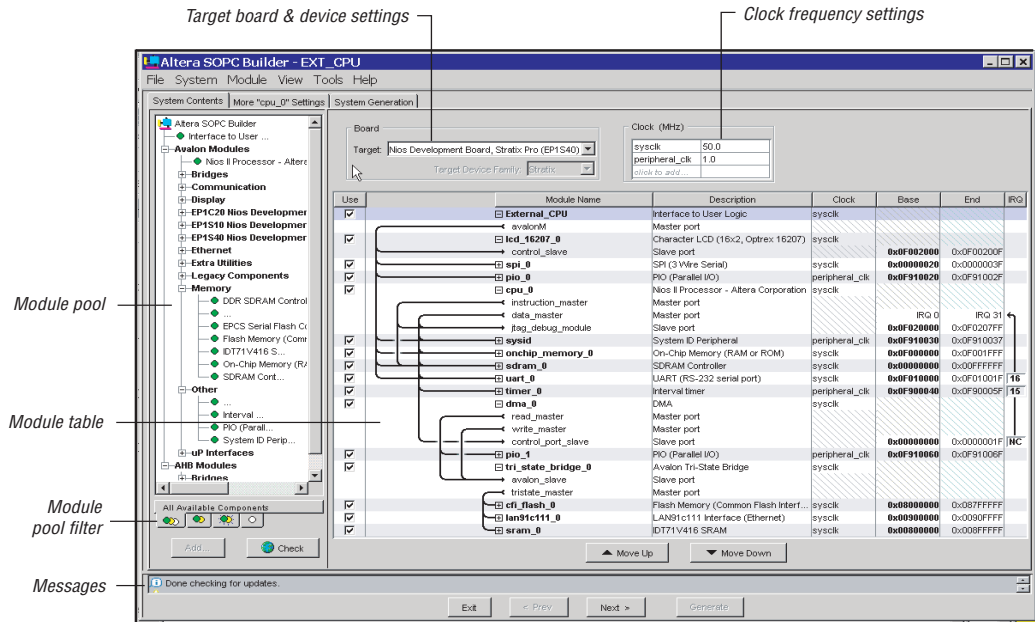
SOPC Builder provides an easy-to-use, table-oriented user interface for defining systems. SOPC Builder contains two primary pages, **System Contents** page and **System Generation** page.

In addition, certain peripherals may add system dependency pages that allow you to specify refinements and relationships with other peripherals in the system.

System Contents Page

You select components and define the system on the **System Contents** page. The page is split into two sections: the module pool and the module table. The module pool lists all the available components, while the module table displays the components added to the current system. **Figure 2-2** shows an example of the **System Contents** page.

Figure 2-2. Systems Content Page



Module Pool

The module pool shows the available library of components organized according to category. Each component appears with a colored dot next to its name. The colors of the dots have the following meanings:

- *Green dot*—Indicates components that are fully-licensed
- *Yellow dot*—Indicates components available for evaluation in some limited form, typically subject to a hard time out or reduced functionality
- *White dot*—Indicates SOPC Builder Ready components available from Altera or partner vendors that are not currently installed on the PC. Evaluation versions of these components can be downloaded from the web for free

You can use the module pool to filter the display for available components, installed components, components available on the web, or components with updates available on the web. In addition, if you have an Internet connection, the module pool is updated to show new components from Altera and partner vendors as they become available.

Module Table

The module table lists components that are included in the current system. Additionally, the module table describes the following elements:

- Connectivity between master and slave ports
- Addresses for each slave port
- Interrupt controller (IRQ) assignments for each slave port
- Arbitration priorities for slave ports shared by multiple-master ports

If **Show Master Connections** (View menu) is turned on, the left side of the module table displays the connectivity between master ports and slave ports. You can selectively connect any master port to any slave port. Whenever two or more master ports share (i.e., have access to) the same slave port, SOPC Builder automatically inserts an arbiter to grant access to the slave when simultaneous requests occur.

To view arbitration priorities, choose **Show Arbitration Priorities** (View menu).

You can connect any master port to any slave port if they use the same interface protocol. If they use different interface protocols, they must communicate through a bridge component, such as the AHB-to-Avalon bridge provided with SOPC Builder.



For more information on master/slave connections and arbitration priorities, see *Application Note 184: Simultaneous Multi-Mastering with the Avalon Bus*.

Additional Settings

The **System Contents** page includes additional options as shown in [Table 2-1](#).

Table 2-1. System Contents Page Options	
Option	Description
Target	Select one of the listed development boards. The Target Device Family and Clock fields are automatically populated. Select Unspecified Board to select the Target Device Family and to make Clock settings manually.
Target Device Family (1)	Select the target FPGA device family from the Target list. SOPC Builder takes advantage of the architectural features of a specified device family when generating the system module.
Clock (1)	The Clock setting specifies the clock domains in the system. Each component in the system can use any one of the clock domains. SOPC Builder adds clock-domain crossing circuitry in the Avalon switch fabric, allowing components in different clock domains to communicate. A component can use its assigned clock to generate clock dividers, baud rate generators, etc. SOPC Builder's built-in testbench generator also uses the Clock setting to simulate a clock of the requested frequency.

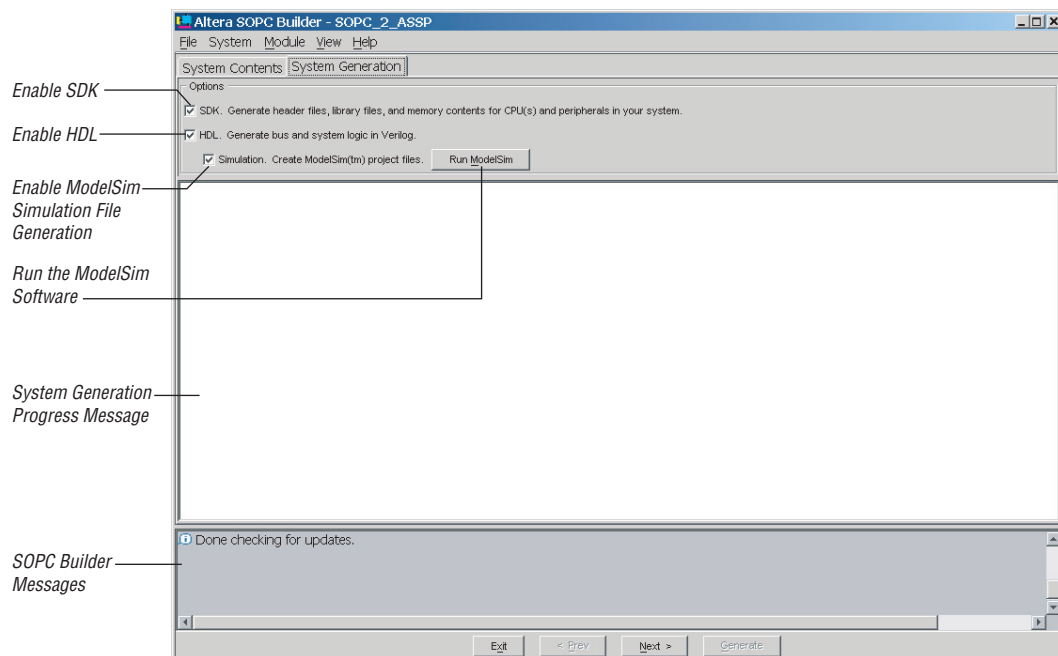
Note to [Table 2-1](#):

- (1) The Quartus II software does not detect this setting automatically. The setting must also be specified in the Quartus II software.

System Generation Page

The **System Generation** page includes settings to control the generation process for the hardware design files, simulation model, and the SDK. [Figure 2-3 on page 2-10](#) shows the **System Generation** page.

Figure 2–3. System Generation Page Note (1)



Note to Figure 2–3:

- (1) Some SOPC Builder Ready processor components may modify the look of this page to allow the user better access to the processor’s software tool chain. Refer to the processor’s documentation for more information.

SDK Option

When the SDK option is turned on, SOPC Builder creates a custom SDK for each CPU in the system. This SDK contains software files (drivers, libraries, and utilities) for any system components that provide software-support files.

You can build software applications for Excalibur™ devices as part of the generation process. The SDK files are arranged into the following directories:

- **inc**—This directory contains header files. These files include the definition for the memory map, register declarations for included peripherals, and macros that can be used in creating embedded software applications.
- **lib**—This directory contains software library files. During system generation, processor components can include commands to have SOPC Builder compile the libraries automatically.

- **src**—This directory provides a location for application source-code development. Example source code files associated with peripherals may also be copied into this directory during system generation.

You should save any file you have edited with a unique filename to prevent the file from being overwritten in a subsequent system generation. Altera recommends that your source code be stored in one of the following locations:

- The **src** directory
- A subdirectory of the **src** directory
- A directory on the same level as the **src** directory in the SDK

You should provide all the SDK files to the software engineers developing application code for the system every time you generate or update the system hardware module.



Certain components, such as the Nios[®] II embedded processor, may modify the contents on the page to provide better access to the software development environment for the processor. Please refer to the CPU component documentation for more information.

HDL Option

To generate hardware description language (HDL) files that describe the system, turn on the **HDL** option. The files are Verilog HDL or VHDL, depending on the language you specify when starting SOPC Builder. The HDL files contain the following components:

- An instance of every component in the system
- The Avalon switch fabric tailored to connect all components in the system. See [“Avalon Switch Fabric” on page 2–4](#) for more details
- A simulation model and a simulation testbench, depending on the **Simulation** option. See the [“Simulation Option”](#) section for more details

Simulation Option

To generate a simulation model and a test bench to simulate a custom model, turn on the **Simulation** option. The testbench is tailored to the structure of system module. The testbench provides the following functionality:

- Instantiates the system module
- Drives clock and reset inputs with default behaviors
- Instantiates and connects the simulation models provided for any components external to the system module (e.g., memory models)

There are custom simulation files associated with each peripheral. SOPC Builder copies these files into a simulation directory during system generation. The simulation directory is separate from the directory that contains the synthesizable HDL.

Simulating with ModelSim

SOPC Builder generates a ModelSim project directory that includes the following files:

- Simulation data files for all memory components that have initialized contents
- A **setup_sim.do** file that contains setup information and aliases customized for simulating the system module
- A **wave_presets.do** file that displays an initial set of bus interface waveforms
- A ModelSim Project File (**.mpf**) for the current system

Run the ModelSim software from within SOPC Builder by clicking **Run ModelSim**. If the ModelSim software is not in your search path, specify the location in the **SOPC Builder Setup** dialog box (File menu).

Simulating with Other Simulators

You can use the SOPC Builder-generated simulation files with simulation software other than ModelSim. However, you cannot use the files generated for ModelSim (**.tcl**, **.do**, **.mpf**, etc.) directly. You can inspect these files and use them as a basis for setting up similar capabilities in other simulation tools.

System Dependency Pages

Certain components, such as a CPU like the Nios embedded processor, display an additional page called a system dependency page in SOPC Builder. System dependency pages display additional parameters or associations to be specified for a component. For example, you can specify the relationship between a CPU and memory components to

indicate which memory is to be used as the program memory and which is to be used as data memory. For components that use system dependency pages, a separate system dependency page is displayed for each instance of the component added to a system.

Generating a System

After you have specified the component and selected the generation options, click the **Generate** button to cause SOPC Builder to generate the system. This button is available from any page in the SOPC Builder user interface.

As the generation process proceeds, SOPC Builder displays messages and information in the system generation progress messages box. SOPC Builder displays the message “Generation Complete” when it is finished, and places a log file in the root directory of the project.

Further Information

For more information and literature on SOPC Builder, visit www.altera.com/sopcbuilder.

