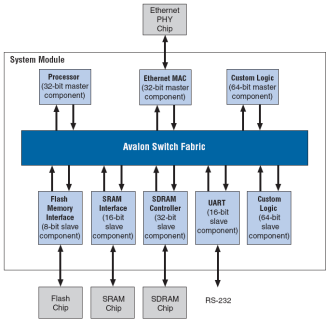
Digital Systems Design

SOPC Builder and NIOS II Hardware Development

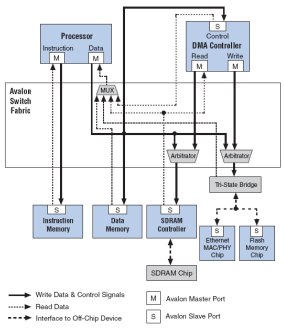
Architecture of SOPC Builder Systems

- An SOPC Builder *component* is a design module that SOPC Builder recognizes and can automatically integrate into a system
- SOPC Builder connects multiple components together to create a top-level HDL file called the *system module*
- SOPC Builder generates *Avalon switch fabric* that contains logic to manage the connectivity of all components in the system

Example of a System Module Generated by SOPC Builder



Avalon Switch Fabric Block Diagram – Example System



SOPC Builder Components

- SOPC Builder components are the building blocks of the system module
- SOPC Builder components use the Avalon interface for the physical connection of components
- SOPC Builder is used to connect any logical device (either on-chip or off-chip) that has an Avalon interface
- The Avalon interface uses an address-mapped read/write protocol that allows master components to read and/or write any slave component

SOPC Builder Components (continued)

- A component can be a logical device that is entirely contained within the system module, such as a processor
- A component can act as an interface to an off-chip device, such as an SRAM interface component
- In addition to the Avalon interface, a component can have other signals that connect to logic outside the system module
- Non-Avalon signals can provide a special-purpose interface to the system module, such as the Ethernet MAC

User Defined Components

- SOPC Builder provides a method to develop and connect user-defined components
- With the Avalon interface, user-defined logic need only adhere to a simple interface based on address, data, read-enable, and write-enable signals
- Design flow to integrate custom logic into an SOPC Builder system:
  - Define the interface to the user-defined component
  - If the component logic resides on-chip, write HDL files describing the component in either Verilog HDL or VHDL
  - Use the SOPC Builder component editor wizard to specify the interface and optionally package your HDL files into an SOPC Builder component
  - Instantiate your component in the same manner as other SOPC Builder Ready components

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 signals and logic that connects master and slave components, including address decoding, data-path multiplexing, wait-state generation, arbitration, interrupt controller, and data-width matching
- SOPC Builder generates the Avalon switch fabric automatically, so that you do not have to manually perform the tedious, error-prone task of connecting hardware modules.

Functions of SOPC Builder

- **Defining & Generating the System Hardware**
  - GUI-based
- **Creating a Memory Map for Software Development**
- **Creating a Simulation Model & Testbench**

Defining & Generating System Hardware

- SOPC Builder GUI allows for defining the structure of a hardware system and generating the system
- The GUI is designed for the tasks of adding components to a system, configuring the components, and specifying how they connect together
- SOPC Builder generates the Avalon switch fabric and output the HDL files that describe the system

Defining & Generating System Hardware (continued)

- During system generation, SOPC Builder outputs the following items:
  - An HDL file for the top-level system module and for each component in the system
  - A Block Symbol File (.**bsf**) representation of the top-level system module for use in Quartus II Block Diagram Files (**.bdf**)
  - (Optional) Software files for embedded software development, such as a memory-map header file and component drivers
  - (Optional) Testbench for the system module simulation project files
- After you generate the system module, it can be compiled directly by the Quartus II software, or instantiated in a larger FPGA design

Creating a Memory Map for Software Development

- For each microprocessor in the system, SOPC Builder optionally generates a header file that defines the address of each slave component
- In addition, each slave component can provide software drivers and other software functions and libraries for the processor
- The process for writing software for the system depends heavily on the nature of the processor in the system
  - For example, NIOS II processor systems use NIOS II processor-specific software development tools
  - These tools are separate from SOPC Builder, but they do use the output of SOPC Builder as the foundation for software development

SOPC Builder (Getting Started)

- One of the easiest ways to get started using SOPC Builder is to read the *NIOS II Hardware Development Tutorial*
  - Step-by-step guide for building a microprocessor system, including CPU, memory, and peripherals

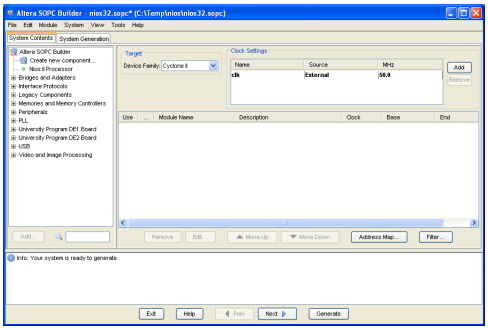
Starting SOPC Builder

- Each SOPC Builder system is associated with one Quartus II project
- Therefore, to launch SOPC Builder, you must first open a project in the Quartus II software
  - Create a new Quartus II project (**rpds17** for example)
  - Create a top level BDF file for the design (**rpds17.bdf**)
  - Import pin assignments from **de2.qsf** from the DVD
- If an SOPC Builder system does not exist in the current Quartus II project directory, SOPC Builder will display the **Create New System** dialog and prompt you to specify:
  - A name for the new SOPC Builder system – **nios32** for example
  - Target HDL – This setting determines the output language of the system module

Starting SOPC Builder

- SOPC Builder saves files in the same directory as the Quartus II project
- Each SOPC Builder system is represented by a file named **<***system module name***>.sopc**, which describes the structure of the system and other system-specific details
- In a purely mechanical sense, the SOPC Builder GUI is a **.sopc** file editor

Elements of the SPOC Builder System



Device Programming

- After System Generation, the output HDL files must be:
  - Compiled into a programming file (*.pof or *.sof) or
  - Integrated into a larger design for subsequent compilation
    - This is the approach Hamblen uses in textbook examples
    - A *.bsf file describing the system (NIOS II in our case) is integrated with other symbols in a schematic design file for compilation and subsequent download to the device
      - Other typical components to include would be a PLL component
      - Ports/pins (IN and OUT) for device pin assignments