Digital Systems Design

Custom Components for NIOS II Systems

Qsys Components

- A Qsys component includes the following elements:
  - Information about the component type, such as name, version, and author.
  - HDL description of the component's hardware.
  - Description of the component interface hardware, such as the types of I/O signals.
  - Description of the parameters that configure the operation of the component.
  - A parameter editor for configuring an instance of the component in Qsys.

Qsys Component Interface Types (subset)

- **Memory-Mapped** -- For Avalon-MM masters and slaves that communicate using memory-mapped read and write commands.
- **Interrupts** -- For point-to-point connections between interrupt senders that generate interrupts, and interrupt receivers that service interrupts.
- **Clocks** -- For point-to-point connections between clock sources and clock sinks.
- **Resets** -- For point-to-point connections between reset sources and reset sinks.
- **Conduits** -- For point-to-point connections between conduit interfaces. You can use the conduit interface type to define a custom collection of signals that do not fit into any other interface category.
  - Conduits are also used to export signals from a component to the top-level design.

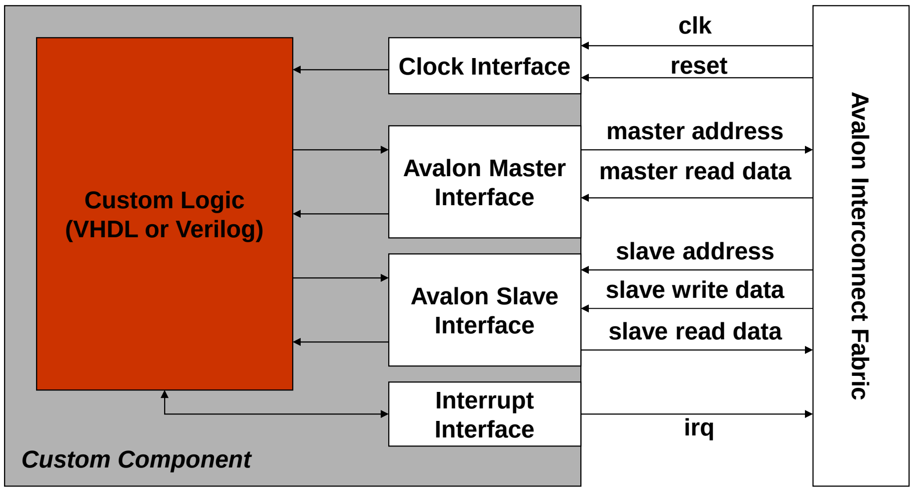
Qsys Component Editor

- The component editor provides a GUI to support the creation and editing of the Hardware Component Description File (**_hw.tcl**) file that describes a component to Qsys.
- Used for:
  - Specifying the HDL file(s) that describe your component
  - Conversely, creating an HDL template for a component by first defining its interface
  - Specifying the signals for each of the component's interfaces, and define the behavior of each interface signal
  - Specifying relationships between interfaces, such as determining which clock interface is used by a slave interface

Component Hardware Structure

- The component editor creates components with the following characteristics:
  - A component has one or more interfaces including
    - Avalon memory mapped (MM) master and/or slave
    - Interrupt sender and receiver
    - Clock input and output
    - Conduit (for exporting signals to the top level)
  - Each interface is comprised of one or more signals
  - The component can represent logic that is instantiated inside Qsys, or can represent logic outside the system with an interface to it

Example Custom Component Structure



Component Editor Functionality

- Within Qsys, the component editor presents several tabs that group related settings
  - Component Type Tab
    - Allows you to specify basic information about the component
      - Name, version number, etc.
  - Files Tab
    - Allows you to create a Qsys component from existing Verilog HDL or VHDL files (Bottom Up Design)
    - or to create an HDL template in either Verilog HDL or VHDL for a Qsys component by first specifying its interfaces (Top Down Design)

Bottom Up Design

- With the **Files** tab you can specify Verilog HDL or VHDL files that describe the component logic
- The component editor analyzes the file by invoking the Quartus II Analysis and Elaboration module
- The component editor analyzes signals and parameters declared for all modules in the top-level file
- If the file is successfully analyzed, the component editor's **Signals & Interfaces** tab lists all design modules in the **Top Level Module** list
  - Will require some care in providing the expected signals within your HDL file
  - A standard naming convention will assist in the analysis

Signals & Interfaces Tab

- The **Signals & Interfaces** tab specifies the purpose of each signal on the top-level component module
  - If you specified a file on the **HDL Files** tab, the signals on the top-level module appear on the **Signals & Interfaces** tab
- Each signal must belong to an interface and be assigned a legal signal type for that interface
- Each signal is assigned to an interface using the **Interface** list

Naming Signals for Automatic Type and Interface Recognition

- The component editor recognizes signal types and interfaces based on the names of signals in the source HDL file, if they conform to the following naming conventions:
- Signal associated with a specific interface:
  - *<interface type>_<interface name>_<signal type>*[_n]
  - For example, a signal with the name *csi_clock_clk* would be of interface type *csi* (clock signal input), in the interface named *clock*, with signal type *clk*
  - The <*signal type*> must match one of the valid signal types for the type of interface
  - _n corresponds to an active low signal
- For any value of <*interface name*> the component editor automatically creates an interface by that name, if necessary, and assigns the signal to it

Valid Values for <Interface Type>

Interface Prefix	Interface Type
asi	Avalon-ST sink (input)
aso	Avalon-ST source (output)
avm	Avalon-MM master
avs	Avalon-MM slave
axm	AXI master
axs	AXI slave
coe	Conduit
csi	Clock Sink (input)
cso	Clock Source (output)
inr	Interrupt receiver
ins	Interrupt sender
ncm	Nios II custom instruction master
ncs	Nios II custom instruction slave
rsi	Reset sink (input)
rso	Reset source (output)
tcm	Avalon-TC master
tcs	Avalon-TC slave

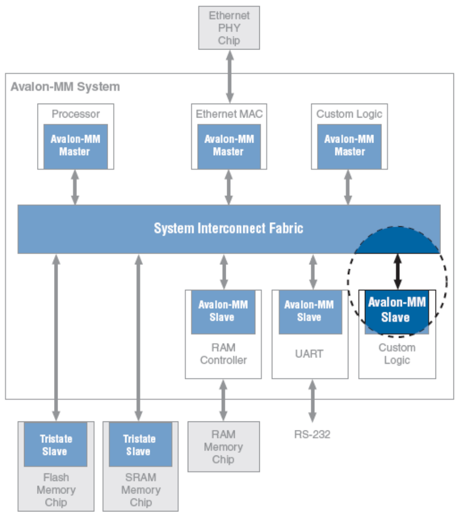
Clock Interfaces

- A clock input interface is used to provide synchronization and reset control for a component
- A typical component has a clock input to provide a timing reference for other interfaces and internal logic
- Valid signal types for a clock interface are:

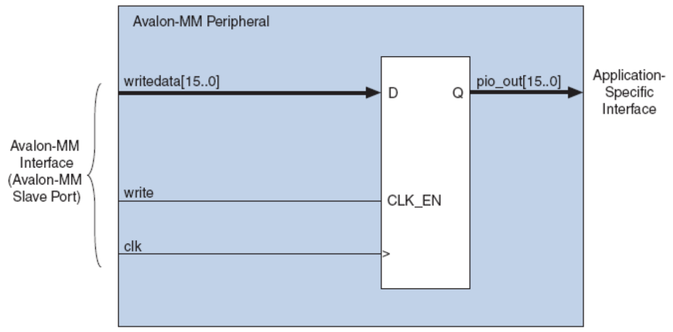
Signal Type	Width	Direction	Required	Description
`clk`	1	Input	No	A clock signal. Provides synchronization for internal logic and for other interfaces.
`reset` `reset_n`	1	Input	No	Reset input. Resets the internal logic of an interface or component to a determined state. `reset` is synchronized to the clock input in the same interface.

Avalon Memory-Mapped Interfaces

- Avalon Memory-Mapped (Avalon-MM) interfaces are used for read/write interfaces on master and slave components in a memory mapped system
- These components include microprocessors, memories, UARTs, and timers, and have master and slave interfaces connected by a system interconnect fabric

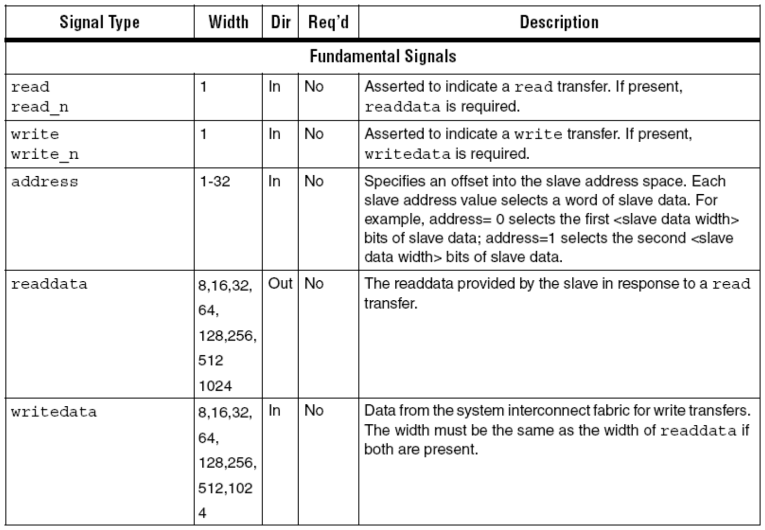


Example Slave Component


Avalon-MM Peripheral
writedata[15..0]
D
Q
pio_out[15..0]
Application-Specific Interface
Avalon-MM Interface (Avalon-MM Slave Port)
write
CLK_EN
clk

- Avalon-MM components typically include only the signals required for the component logic
- The 16-bit general-purpose I/O peripheral shown only responds to write requests, therefore it only includes the slave signals required for write transfers

Slave Port Signals (subset of those actually available)



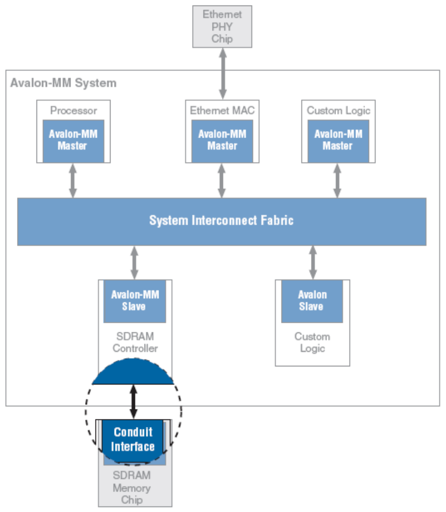
Signal Type	Width	Dir	Req'd	Description
Fundamental Signals				
`read` `read_n`	1	In	No	Asserted to indicate a `read` transfer. If present, `readdata` is required.
`write` `write_n`	1	In	No	Asserted to indicate a `write` transfer. If present, `writedata` is required.
`address`	1-32	In	No	Specifies an offset into the slave address space. Each slave address value selects a word of slave data. For example, address= 0 selects the first <slave data width> bits of slave data; address=1 selects the second <slave data width> bits of slave data.
`readdata`	8,16,32, 64, 128,256, 512 1024	Out	No	The readdata provided by the slave in response to a `read` transfer.
`writedata`	8,16,32, 64, 128,256, 512,1024	In	No	Data from the system interconnect fabric for write transfers. The width must be the same as the width of `readdata` if both are present.

Interfaces Tab

- The **Interfaces** tab allows you to configure the interfaces on your component and specify a name for each interface
- Interface name
  - Identifies the interface and also appears in the Qsys connection panel
  - Also used to uniquely identify any signals that are ports on the top-level Qsys system
- The **Interfaces** tab also allows you to configure the type and properties of each interface
  - For example, an Avalon-MM slave interface has timing parameters that you must set appropriately

Conduit Interfaces

- Conduit interfaces are used to group together an arbitrary collection of signals to be exported to the outside of a Qsys system
- A conduit interface can consist of both input and output signals



Conduit Signals

Signal Type	Width	Direction	Required	Description
`export`	*n*	In, out or bidirectional	Yes	A conduit interface consists of one or more signals of arbitrary width, that are inputs or output, of type `export`. All of these signals are exported out the top level of the SOPC Builder system.

- ***coe_myoutputsignal_export*** would be an example conduit signal name that could be recognized as a signal within Qsys