Avalon-MM Slave Template

Date: 09/08/2008

Disclaimer

These component templates may be used within Altera® devices only and remain the property of Altera. They are being provided on an "as-is" basis and as an accommodation, and therefore all warranties, representations, or guarantees of any kind (whether express, implied or statutory) including, without limitation, warranties of merchantability, non-infringement, or fitness for a particular purpose, are specifically disclaimed. Altera expressly does not recommend, suggest, or require that these examples be used in combination with any other product not provided by Altera.

If you encounter an issue or have an enhancement request, please log onto mySupport to file a service request.

Overview

Provided in this package is a slave template that you can use to make your own custom logic accessible from within SOPC Builder. Signals are exposed at the top of the system so that you can connect them to your custom logic. You can also use the HDL provided in the template to add new capabilities to an existing SOPC Builder component. The component provides the following functionality:

- Pipelined read and write capabilities
- Up to 16 individual read and write registers
- Optional output loopback mode to make outputs readable to software
- Data widths of 8, 16, 32, 64, 128, 256, 512, and 1024
- Optional synchronization logic that you can use for handshaking purposes with your own custom hardware

Installation

In order to use the templates, simply extract the 'ip' directory into your own hardware project directory. SOPC Builder references the ip directory by default. When you open SOPC Builder the slave template will appear in the component listing on the left side of the SOPC Builder user interface. The master templates will appear in the group called "Templates". You will be asked to setup parameters such as the data width while adding the slave to the system. Refer to the HDL port listing section of this document to learn more about using these settings.

Slave Block Diagram

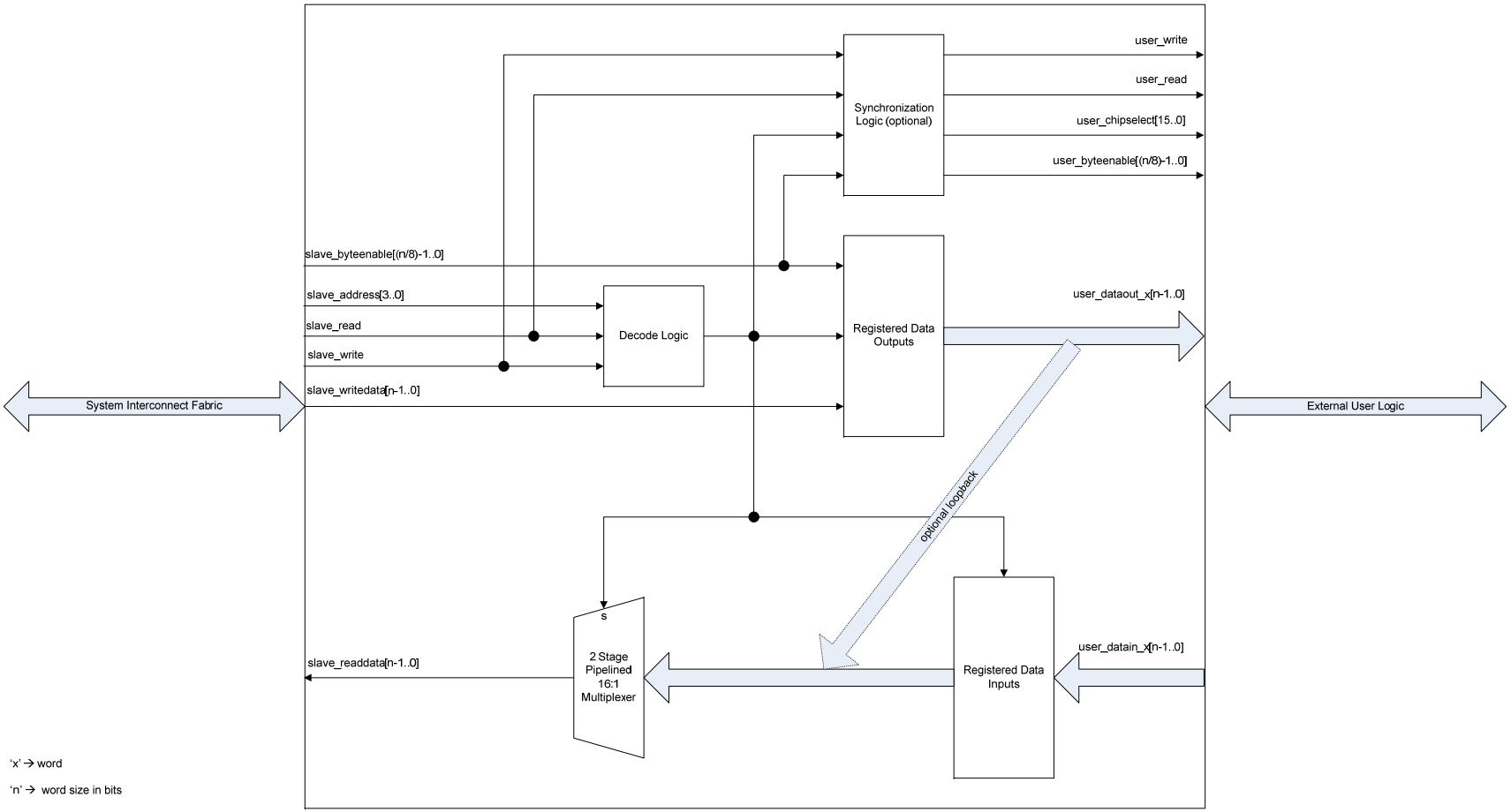

Figure 1. Slave Template

Usage

The template has multiple modes which are set using HDL parameterization. Most of the settings do not directly affect the hardware generated and are used by the elaboration callback provided in the tcl file to stub input or output signals. By stubbing the signals registers will be optimized away resulting in a logic savings. When the component is setup for output with loopback the user input signal is ignored and the output register is directly looped back as an input.

One of the key differentiators between this slave template and the PIO component available in SOPC Builder is the ability to synchronize the inputs and outputs to the external surrounding logic. Typically with a PIO component it is written or read but the logic connected to it has no way of knowing when these operations occur. When you enable the synchronization signals in the template you can use them to determine when an input is being read or an output is being written to. To determine if a write operation is occurring you should use user_chipselect, user_write, and user_byteenable to identify which register and byte lanes are being written to. To determine if a read operation is occurring you should use user_chipselect, and user_read to identify which register is being read.

Signal Name	Direction	Width	Usage
user_dataout_x	Output	Datawidth (default 32)	Output signal to connect to your own custom hardware
user_datain_x	Input	Datawidth (default 32)	Input signal to connect to your own custom hardware
user_chipselect[1]	Output	16	Each chip select corresponds to an I/O pair. If you attempt to accesses an I/O that is disabled the chip select will still assert.
user_byteenable[1]	Output	Datawidth / 8	Byte enables used during write cycles, should be ignored for read cycles
user_write[1]	Output	1	
user_read[1]	Output	1	

Table 1. User Interface Signals

Note: 'x' represents the I/O pair number, this value varies from 0 to 15. [1] denotes signals that are optional.

The Avalon-MM slave port that is connected to the system interconnect fabric supports dynamic bus sizing allowing individual byte lanes to be accessed. This also allows narrow masters to write to this component if the data width is larger. Since the slave port never blocks the waitrequest signal is not necessary and a fixed read latency of three clock cycles is used instead.

Signal Name	Direction	Width	Usage
slave_address	Input	4	Word address
slave_writedata	Input	Datawidth (default 32)	Write data (to your custom logic)
slave_readdata	Output	Datawidth (default 32)	Read data (from your custom logic)
slave_write	Input	1	Write
slave_read	Input	1	Read
slave_byteenable	Input	Datawidth / 8	Byte enables used for write cycles to identify byte lanes to be written to in the word
clk	Input	1	All logic is synchronized to this clock
reset	Input	1	Reset used to clear all registers

Table 2. Slave and Clock Interface Signals

HDL Port Listings

The following port listing is for the provided HDL file. The only interface exposed to your logic is the user interface which contains signals with “user_” as a prefix. When you insert the provided slave template into your system you will be prompted to enter parameterization values. You can also add the master HDL to your own SOPC Builder component and assign the signals and interfaces manually.

```
module slave_template (
     clk,
     reset,
     slave_address,
     slave_read,
     slave_write,
     slave_readdata,
     slave_writedata,
     slave_byteenable,
     user_dataout_0,
     user_dataout_1,
     user_dataout_2,
     user_dataout_3,
     user_dataout_4,
     user_dataout_5,
     user_dataout_6,
     user_dataout_7,
     user_dataout_8,
     user_dataout_9,
     user_dataout_10,
     user_dataout_11,
     user_dataout_12,
     user_dataout_13,
     user_dataout_14,
     user_dataout_15,
     user_datain_0,
     user_datain_1,
     user_datain_2,
     user_datain_3,
     user_datain_4,
     user_datain_5,
     user_datain_6,
     user_datain_7,
     user_datain_8,
     user_datain_9,
     user_datain_10,
     user_datain_11,
```

```
      user_datain_12,
      user_datain_13,
      user_datain_14,
      user_datain_15,
      user_chipselect,
      user_byteenable,
      user_write,
      user_read
);
```

Parameter	Range	Usage
DATA_WIDTH	8, 16, 32, 64, 128, 256, 512, 1024 (default 32)	Data path width
ENABLE_SYNC_SIGNALS	0 or 1	Set to 1 to enable the synchronization signals “user_chipselect”, “user_byteeable”, “user_write”, “user_read”, 0 to disable the optional signals
MODE_X ‘X’ 0 though 15	0-4 (default 2)	0 = output only 1 = input only 2 = output and input 3 = output with loopback 4 = disabled

Table 2. Slave Parameterization Values

Template

A file called “slave_template.v” has been provided which contains all the logic for the slave template. Signals that are not required are stubbed before they are connect to the system interconnect fabric. This is provided by the elaboration callback provided in the component tcl file called “slave_template_hw.tcl”. This tcl file allows you to make setting changes complete with validation and automatically derived settings. Since all the validation is provided you do no need to worry about putting the component into an unsupported configuration.

To learn more about callbacks and the component interface tcl scripting refer to the following chapter of the Quartus II handbook, Volume 4 SOPC Builder “Component Interface Tcl Reference”