OpenCPI
CDC Single Bit Tester Component Data Sheet

OpenCPI Release: v2.4.7

Revision History

Revision	Description of Change	Date
v1.6	Initial Release	02/2020

Summary - CDC Single Bit Tester

Name	cdc_single_bit_tester
Worker Type	Application Worker
OpenCPI Release	v2.4.7
Last Update	02/2020
Component Library	ocpi.assets.misc_comps
Workers	cdc_single_bit_tester.hdl
Tested Platforms	isim, xsim, ZedBoard(PL)

Functionality

The cdc_single_bit_tester serves as a testbench for the `single_bit` cdc primitive. The `single_bit` primitive synchronizes a single-bit from the source clock domain to the destination clock domain. The output of the synchronizer is sent to worker's output port.

For normal operation, the input signal must be sampled two or more times by the destination clock. To achieve this requirement when synchronizing from:

a) Slow Signals into Fast Clock Domain:
The destination clock must be >= 2x frequency of the source clock.
b) Fast Signals into Slow Clock Domain:
The minimum input signal width must be 2x the period of the destination clock and the input pulses must be separated by 2x src_clk cycles to ensure proper crossing of the CDC boundary. Depending on the phase and frequency relationship of the source and destination clocks, the generated pulse in the destination domain may be +1 cycle more than the input pulse.

See `http://www.sunburst-design.com/papers/CummingsSNUG2008Boston_CDC.pdf` - Section 4.4 and `https://m.eet.com/media/1137372/17561-310388.pdf` for more information on the 2x requirement.

The number of register stages used in the synchronizers is configurable. An optional input register may be used to register the input signal in source clock domain prior to being synchronized. When possible, it is recommended (default) to register the input signal in the source clock domain before sending it across the clock domain crossing (CDC) into synchronizers.

Worker Implementation Details

The cdc_single_bit_tester worker uses a LFSR to generate data and the bit shifted out from LFSR is then used as the input to the `single_bit` primitive. Once the input signal has crossed into the destination clock domain it is stored in a FIFO until the worker's output port is ready for the data. The FIFO is used because the data is sent only once and also used to mitigate synchronizing the backpressure from the output port to the LFSR.

The `src_clk_hz` and `dst_clk_hz` parameter properties are used to define the source and destination domain clock frequencies.

A clock generator is used to generate the clocks for the source and destination clock domains when the source and destination clock domains are different.

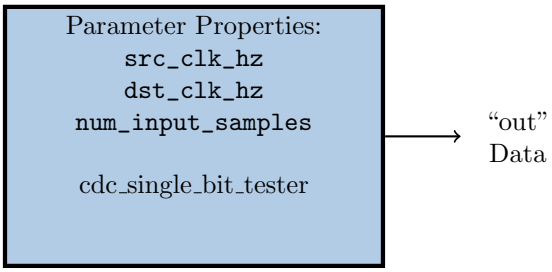
Fast clock domain to slow clock domain:
When the source clock frequency is faster than destination clock frequency, the destination domain reset signal is sent through a reset synchronizer to synchronize it to the source domain. The source domain uses the sychronized reset to enable a advance_counter module that outputs a signal to enable the LFSR. It is used to ensure that the minimum input signal width to the `single_bit` primitive is 2x the period of the destination clock and they are seperated by 2x src_clk cycles to ensure proper crossing of the CDC boundary. The FIFO is enabled when the destination clock domain has come out of reset.

Slow clock domain to fast clock domain:
When the source clock frequency is slower than the destination clock frequency, the LFSR are enabled when the source domain has come out of reset. The source domain reset signal is sent through a reset synchronizer to synchronize it to the destination domain and is used to enable the FIFO.

Same clock domain:
When the source clock frequency is equal to the destination clock frequency, the destination domain reset signal is sent through a reset synchronizer to synchronize it to the source domain. The source domain uses the sychronized reset to enable a advance_counter. The source domain reset signal is sent through a reset synchronizer to synchronize it to the destination domain and is used to enable the FIFO.

Block Diagrams

Top level



Source Dependencies

cdc_single_bit_tester.hdl

- assets/components/misc_comps/cdc_single_bit_tester.hdl/cdc_single_bit_tester.vhd
- core/hdl/primitives/cdc/single_bit.vhd
- core/hdl/primitives/cdc/cdc_pkg.vhd

Component Properties

Name	Type	SequenceLength	ArrayDimensions	Accessibility	Valid Range	Default	Description
src_clk_hz	Float	-	-	Parameter	Standard	100000000.0	Source clock frequency
dst_clk_hz	Float	-	-	Parameter	Standard	100000000.0	Destination clock frequency
num_input_samples	Uchar	-	-	Parameter	Standard	15	Number of input samples sent.

Worker Properties

cdc_single_bit_tester.hdl

Name	Type	SequenceLength	ArrayDimensions	Accessibility	Valid Range	Default	Description
src_clk_hz	Float	-	-	Parameter	Standard	100000000.0	Source clock frequency
dst_clk_hz	Float	-	-	Parameter	Standard	100000000.0	Destination clock frequency
num_input_samples	Uchar	-	-	Parameter	Standard	15	Number of input samples sent.

Component Ports

Name	Producer	Protocol	Optional
out	True	(none)	False

Worker Interfaces

cdc_single_bit_tester.hdl

Type	Name	Producer	Protocol	Optional	DataWidth	Clock	ClockDirection	WorkerEOF	InsertEOM
StreamInterface	out	True	(none)	False	32	-	out	True	True

Data Timing and Signals

The cdc_single_bit_tester worker uses the clock from the Control Plane as input to the clock generator. The output of the clock generator is used to drive the output port clock.

Performance and Resource Utilization

Test and Verification

There are files used to validate the data the cdc_single_bit_tester worker sends. The files used in the verification were created from the output of the unit test when running the unit test in simulation and contain the expected output data when there is no metastability. Since there might or might not be a perfect match of the output data to the expected output data when running the tests in hardware due to metastability or phase offset, the output data is correlated with the expected output data. If there is at least a 0.7 positive Pearson product-moment correlation coefficient, then test case is considered passing. This correlation is only used when running the unit test on hardware. If the test are run in simulation, the output data is compared with expected output data. If the they match exactly, then test case is considered passing.