XAPP562 (v1.1.1) April 20, 2007

Configurable LocalLink CRC Reference Design

Author: Nanditha Jayarajan

Summary

The Cyclic Redundancy Check (CRC) is a checksum technique for testing data reliability and correctness. This application note shows how to implement *Configurable CRC Modules* with LocalLink interfaces. Users tailor the module features to suit the protocol or application implemented in their system. The user-specified options for each of the configurable features are input parameters to the VHDL code for the modules. The VHDL source files for the CRC modules are coded using generate statements. The modules have two LocalLink interfaces: an *upstream* interface (US) and a *downstream* interface (DS)

A module connected to the upstream interface is the source of the data. The CRC module passes the data from the US interface onto the DS interface. This allows the CRC modules to be inserted into the data path seamlessly (with consideration for latency).

Refer to *IEEE 802.3 Cyclic Redundancy Check* (XAPP209) for more details on CRC and an alternate implementation of a configurable CRC module.

Key Features

- Standard LocalLink on upstream and downstream interfaces
- Fully configurable according to compile-time parameters
- Key Parameters:
  - Data Width
  - CRC Polynomial
  - CRC Calculation Parameters
    - Transpose input data bytes
    - Transpose CRC bytes
    - Complement input data
    - Complement CRC
  - Receive CRC Parameters
    - Strip incoming CRC option
  - Transmit CRC Parameters
    - Overwrite pad word(s) with CRC[1]
    - Deassert DST_RDY_N_US while CRC words are being inserted
- Transmit CRC Module
  - Generates CRC over data from US interface using user-specified polynomial
  - Streams US data to DS interface. A fixed latency depends on the CRC polynomial and the data width.
  - Appends calculated CRC to the frame on the downstream interface.

1 This feature is not supported in version 1.1 of the document release.

- Receive CRC Module
  - Generates CRC on data from US interface using user-defined polynomial
  - Can either keep incoming CRC value or strip it on downstream interface
  - Indicates validity of incoming CRC from US interface using two out-of-band signals

Example Module Usage

Figure 1 shows an example use of the CRC modules with a MAC or other streaming protocol block.

All the port names for the CRC Transmit and Receive modules have been divided into two parts, upstream and downstream. The upstream interface is the source of the data into the CRC modules, while the downstream interface receives data from the CRC modules. The upstream interface signals have a suffix of _US and the downstream signals have a suffix of _DS.

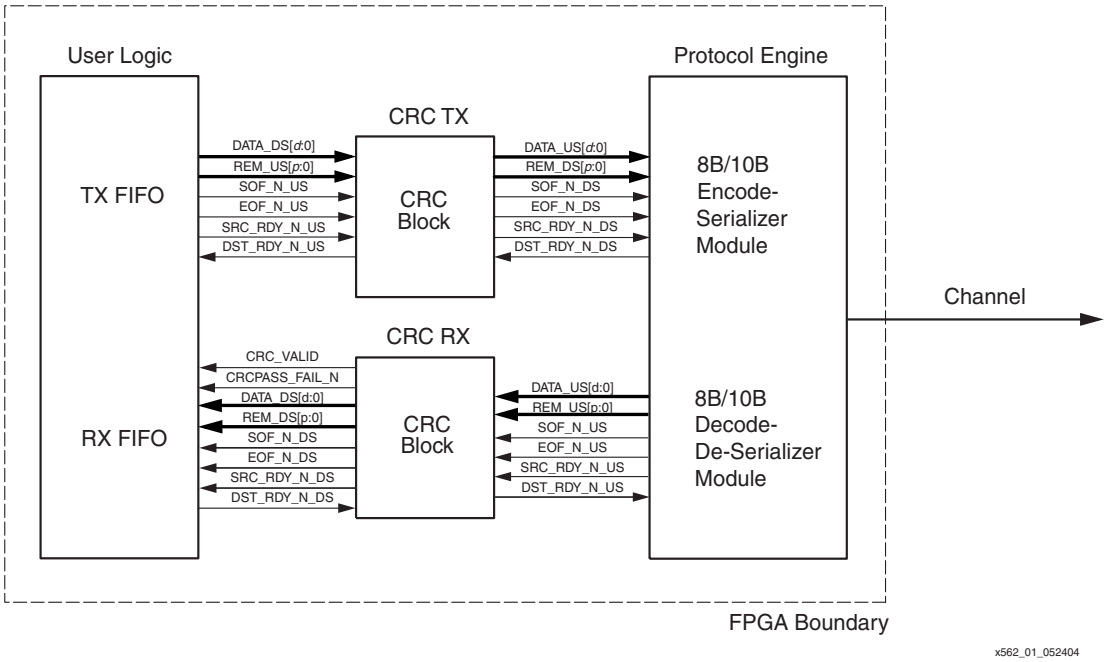

Figure 1: **LocalLink CRC Blocks**

The configurable CRC solution can be broken into two parts. These parts are discussed in subsequent sections of this document:

1. TXCRC – generates and inserts CRC on the transmit path.
2. RXCRC – verifies the CRC on the receive path.

TXCRC Module Functional Description

The TXCRC module has two LocalLink interfaces shown in Figure 2. The left side is the upstream interface, while the right side is the downstream interface. This makes this block both a LocalLink source and destination, which allows this block to be inserted into the transmit path of the system. The module generates a CRC value from data input on the upstream interface. The module then inserts the CRC byte(s) at the end of the frame on the downstream interface.

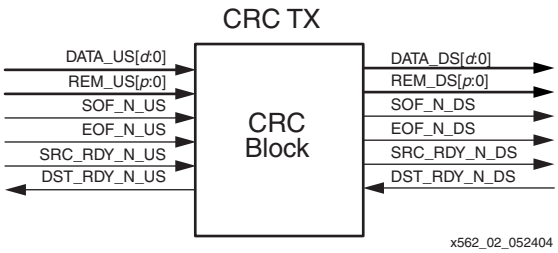

Figure 2: **TXCRC Module**

The user can choose the method of CRC insertion from two options[2]:

1. The user can choose to provide pad Word(s) after the data. The number of pad Words must be equal to the width of the CRC and the last pad Word must coincide with the TX_EOF_N_US. The RX_REM_US must coincide with TX_EOF_N_US and must indicate the number of valid bytes for the frame including the pad Word(s). These pad Word(s) are overwritten by the generated CRC value on the downstream interface[2].
2. TX_DST_RDY_N_US can be used to pause the transfer of data from the upstream module. When the CRC Word(s) are being inserted into the downstream interface, TX_DST_RDY_N_US is deasserted, indicating that the CRC module is not ready to accept new data. After the CRC Word(s) have been inserted, TX_DST_RDY_N_US is asserted.

Refer to "LocalLink CRC Parameters" for more details on these parameters.

A module that uses the TX_DST_RDY_N_US option cannot accept back-to-back frames, because TX_DST_RDY_N_US is deasserted to stop the transfer of data to the TXCRC module at TX_EOF_N_US. Hence, the module is not ready to accept a new frame until TX_DST_RDY_N_US is asserted after all the CRC Word(s) have been inserted.

There is a fixed latency through the TXCRC module. See Table 1 for resource numbers and latency through the modules for various parameter options.

Table 1: **TXCRC Module FPGA Resource Requirements**

Data Width	Aligned Latency = 2 (Slices/LUTs/FFs)	Pipelined Non-Aligned Latency = 3 (Slices/LUTs/FFs)	Non-Pipelined Non-Aligned Latency = 2 (Slices/LUTs/FFs)
128	648/595/315	1283/1836/437	1675/2404/1088
64	372/382/191	1,121/1,775/569	1,048/1,717/242
32	225/251/128	465/678/307	425/640/142
16	152/175/114	231/309/198	198/268/115
8	128/168/103	128/168/103	128/168/103

[2] In version 1.1 of the document release, only the DST_RDY_N_US option is supported.

Table 2 shows the performance numbers and latency for target device: XC2VP7 speed grade = -6.

Table 2: **Design Performance: Target Device XC2VP7 Speed Grade = -6**

Data Width	Non-Pipelined Aligned Latency = 2 (Slices/LUTs/FFs)	Pipelined Non-Aligned Latency = 3 (Slices/LUTs/FFs)	Non-Pipelined Non-Aligned Latency = 2 (Slices/LUTs/FFs)
128	200 MHz	127.3 MHz	103.3 MHz
64	226 MHz	184.5 MHz	135.6 MHz
32	227 MHz	223.9 MHz	180.5 MHz
16	288.85 MHz	252 MHz	227.1 MHz
8	257 MHz	257MHz	257 MHz

Interface Signals for TXCRC Modules

The interface signals for TXCRC modules are defined in Table 3, which continues on the next page.

Table 3: **TXCRC Module Interface Signals**

Name	Direction	Active	Definition
RESET	Input	High	**Reset**: The TXCRC module is reset. The reset is synchronous to CLK.
CLK	Input	N/A	**Clock Input**: All signals are synchronous to this clock.
Upstream LocalLink Interface			
TX_DATA_US[(*d*-1):0]	Input	N/A	**Data Bus**: Packet data is transmitted across this bus. The variable *d* is a multiple of eight. The user specifies this value.
TX_SOF_N_US	Input	Low	**Start of Frame**: Indicates the beginning of a frame transfer on the data bus. This signal is coincident with the first data word. This signal also indicates the start of calculation.
TX_EOF_N_US	Input	Low	**End of Frame**: Indicates the end of a frame transfer. This signal also indicates the end of calculation. This signal is coincident with the pad Word if one is being provided. Otherwise, it is coincident with the last valid data word and DST_RDY_N_US is deasserted during the insertion of the CRC bytes. ***Note***: All data over which the CRC is to be calculated should be within the SOF_N_US and EOF_N_US signals.
TX_SRC_RDY_N_US	Input	Low	**Source Ready**: Indicates data that is provided by the source interface on the data bus is valid during the current cycle.

Table 3: **TXCRC Module Interface Signals *(Continued)***

Name	Direction	Active	Definition
TX_DST_RDY_N_US	Output	Low	**Destination Ready**: Indicates that the destination interface is ready to accept data presented to it on the data bus in the current cycle.
TX_REM_US[p:0](*p*=log2(*d*/8)-1)	Input	Low	**Remainder**: Indicates the byte offset of the last valid byte on the data bus for the last PDU transfer. Remainder value is valid concurrent with end-of-frame assertions. The remainder is binary encoded. If the data width is 32-bit, then the remainder signal is REM[1:0]. Remainder value of 0 indicates byte data [31:24] is valid, while a value of 3 indicates all bytes are valid.
Downstream LocalLink Interface			
TX_DATA_DS[*d* -1:0]	Output	N/A	**Data Bus**: Packet data is transmitted across this bus. The variable *d* is a multiple of eight. A fixed latency is inserted into the data from the upstream interface and is streamed to the downstream interface. The CRC that is calculated over the entire frame is appended to the end of the frame. The amount of latency depends on the width of the CRC and the width of the data bus.
TX_SOF_N_DS	Output	Low	**Start of Frame**: Indicates the start of frame transfer. The upstream TX_SOF_N_US signal is forwarded to the downstream interface with a fixed latency.
TX_EOF_N_DS	Output	Low	**End of Frame**: Indicates the end of a frame transfer including the CRC. The TX_EOF_N_US signal is passed to the downstream interface with a fixed latency.
TX_DST_RDY_N_DS	Input	Low	**Destination Ready**: Indicates that the destination interface is ready to accept data presented to it on the data bus in the current cycle. The DST_RDY_N_DS signal is passed through to the upstream interface without any latency.
TX_SRC_RDY_N_DS	Output	Low	**Source Ready**: Indicates data that is provided by the source interface on the data bus is valid during the current cycle. The SRC_RDY_N_US is passed through to the downstream interface with a fixed latency.
TX_REM_DS[*p*:0]	Output	N/A	**Remainder**: Indicates the byte offset of the last valid byte on the data bus for the last PDU transfer. Remainder value is valid concurrent with end-of-frame assertions. The TX_REM_US is appropriately delayed and modified on the downstream interface.

Data Transfer Through the TXCRC Module

Figure 3 shows the waveforms through the TXCRC block. The calculation of the CRC starts on the assertion of the TX_SOF_N_US signal on the upstream interface. The calculation is paused whenever TX_SRC_RDY_N_US or TX_DST_RDY_N_US is deasserted and resumes when both are asserted.

There is a fixed latency between the upstream and downstream interface. The TX_EOF_N_US signal terminates the calculation of the CRC. CRC is not calculated on any data on the upstream data bus after the TX_EOF_N_US signal. In Figure 3, the data width is assumed to be 32, while the CRC polynomial used is assumed to be CRC32. As shown in Figure 3, the TX_REM_US value coinciding with RX_EOF_N_US is 1. This means that two bytes of data are valid. The four bytes of CRC are appended to the end of the packet, and the REM value is modified and appropriately shifted to coincide with TX_EOF_N_DS. While the CRC bytes are being appended, the TX_DST_RDY_N_US signal is deasserted indicating that the CRC block is not ready to accept new data on the upstream interface.

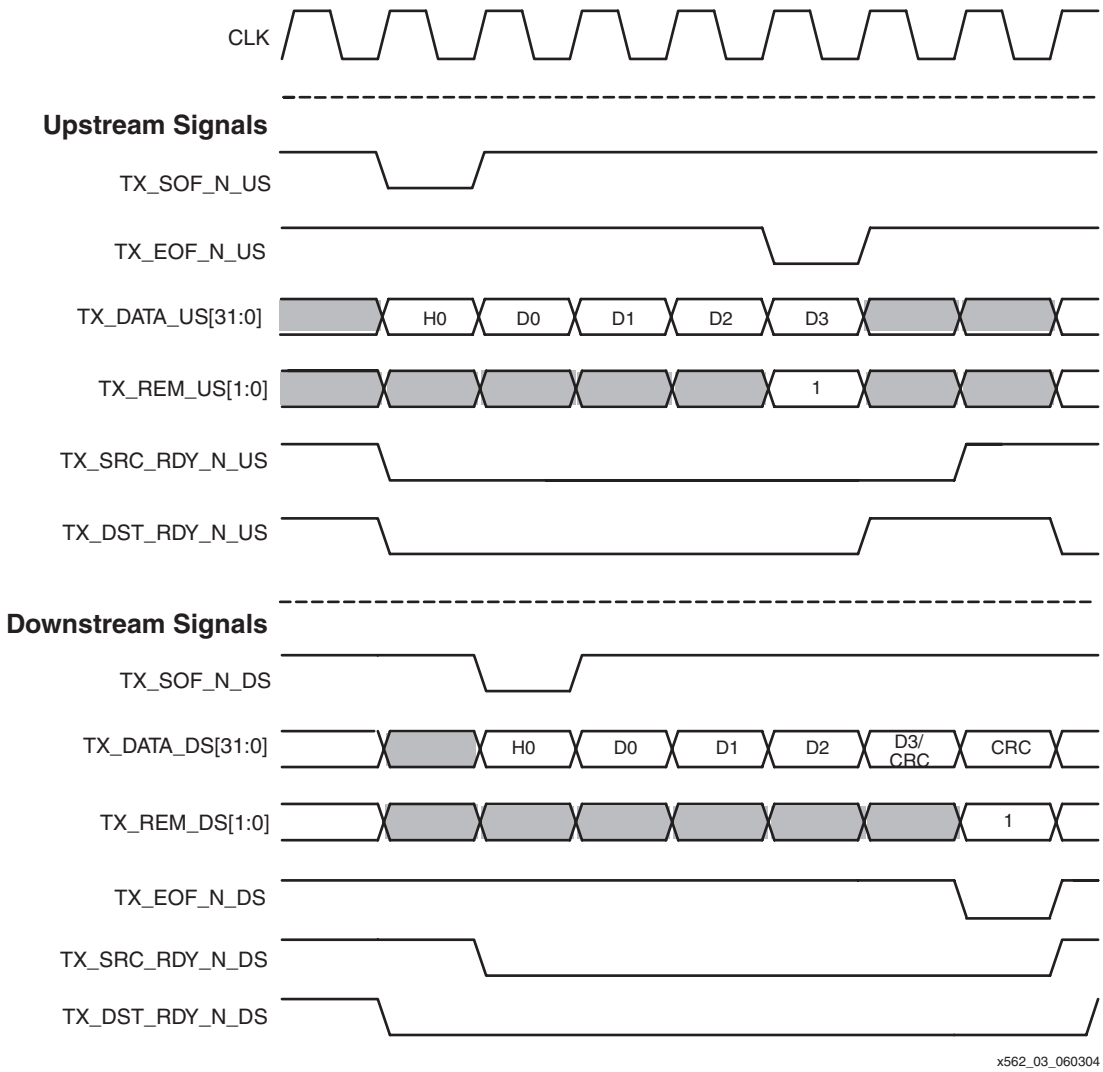

Figure 3: **Frame Transfer Through the TXCRC Block**

RXCRC Module Functional Description

The RXCRC module calculates the CRC on incoming data from the upstream LocalLink interface. It also passes the data through to its downstream LocalLink interface. The downstream interface includes two extra signals, CRCPASS_FAIL_N and CRC_VALID. The CRCPASS_FAIL_N signal indicates whether or not the incoming frame contains a valid CRC. This signal is valid only when CRC_VALID is asserted high. The interface of the RXCRC block is shown in Figure 4.

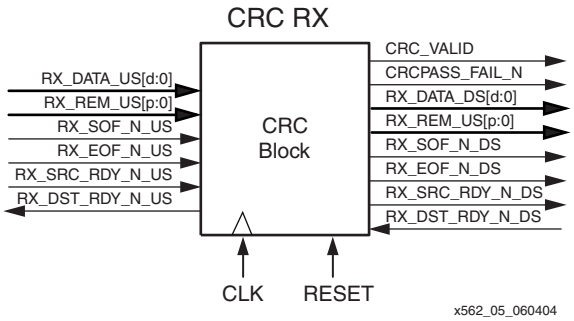

Figure 4: **Module**

The interface on both sides of this block follows the LocalLink Protocol specification (SP006: LocalLink Interface). Click the **Access Lounge** button and follow the registration instructions to download the specification.

The user can choose the frame format in which the incoming data from the upstream interface is passed on to the downstream interface. There are two options:

1. The CRC that is received at the end of the frame is stripped and the raw data is passed on. The RX_EOF_N_DS signal is shifted to indicate the new EOF without the CRC. The CRCPASS_FAIL_N signal indicates whether the packet passes. This signal is valid at CRC_VALID.
2. The data is passed on without removing the CRC. The CRCPASS_FAIL_N signal indicates whether the packet passes. This signal is valid at CRC_VALID.

See Table 4 for RXCRC module resource numbers and latency through the module for various parameter options

Table 4: **RXCRC Module FPGA Resource Requirements**

Data Width	Aligned Latency = 2 (Slices/LUTs/FFs)	Pipelined Non-Aligned Latency = 3 (Slices/LUTs/FFs)	Non-Pipelined Non-Aligned Latency = 2 (Slices/LUTs/FFs)
128	578/439/176	N/A	N/A
64	335/285/267	973/1473/440	875/1352/233
32	201/187/68	387/512/235	347/481/168
16	121/108/61	175/188/146	145/160/61
8	145/160/93	145/160/93	145/160/93

Table 5 shows the performance numbers and latency for target device: XC2VP7 speed grade = -6.

Table 5: **Design Performance: Target Device XC2VP7 Speed Grade = -6**

Data width	Non-Pipelined Aligned Latency = 2	Pipelined Non-Aligned Latency = 3	Non-Pipelined Non-Aligned Latency = 2
128	202 MHz	N/A	N/A
64	223.6 MHz	182 MHz	133 MHz
32	268 MHz	227 MHz	182 MHz
16	295 MHz	287 MHz	191.35 MHz
8	263 MHz	263 MHz	263 MHz

Interface Signals for RXCRC Modules

The interface signals for RXCRC modules are defined in Table 6.

Table 6: **RXCRC Module Interface Signals**

Name	Direction	Active	Definition
RESET	Input	High	**Reset**: The TXCRC module is reset. The reset is synchronous to CLK.
CLK	Input	N/A	**Clock Input**: All signals are synchronous to this clock.
Upstream LocalLink Interface			
RX_DATA_US[(*d*-1):0]	Input	N/A	**Data Bus**: Packet data is transmitted across this bus. The variable *d* is a multiple of 8. The user specifies this value.
RX_SOF_N_US	Input	Low	**Start of Frame**: Indicates the beginning of a frame transfer on the data bus. This signal also indicates the start of calculation to the RXCRC module. This signal is coincident with the pad Word if one is provided.
RX_EOF_N_US	Input	Low	**End of Frame**: Indicates the end of a frame transfer on the data. This signal indicates the end of calculation of the CRC. This signal is coincident with the last Word of the CRC of an incoming frame.
RX_SRC_RDY_N_US	Input	Low	**Source Ready**: Indicates data that is provided by the source interface on the data bus is valid during the current cycle.
RX_DST_RDY_N_US	Output	Low	**Destination Ready**: Indicates that the destination interface is ready to accept data presented to it on the data bus in the current cycle.

Table 6: **RXCRC Module Interface Signals *(Continued)***

Name	Direction	Active	Definition
RX_REM_US[p:0](p=$\log_2$(d/8)-1)	Input	N/A	**Remainder**: Indicates the byte offset of the last valid byte on the data bus for the last PDU transfer. Remainder value is valid concurrent with end-of-frame assertions. If the data width is 32-bit, then the remainder signal is REM[1:0]. Remainder value of 0 indicates byte data[31:24] is valid, while a value of 3 indicates all bytes are valid.
Downstream LocalLink Interface			
RX_DATA_DS[d:0]	Output	N/A	**Data Bus**: Packet data is transmitted across this bus. The variable d is a multiple of eight. A fixed latency is inserted into RX_DATA_US before it is passed on to the downstream interface. The CRC bytes at the end of the packet can be stripped and the raw data passed on, or the entire packet including the CRC bytes can be passed to the downstream interface.
RX_SOF_N_DS	Output	Low	**Start of Frame**: Indicates the start of a frame transfer on the data and CRC. The RX_SOF_N_US signal is passed to the downstream interface with a fixed latency.
RX_EOF_N_DS	Output	Low	**End of Frame**: Indicates the end of a frame transfer including the CRC. The input end-of-frame signal is passed on to the downstream interface with a fixed latency.
RX_DST_RDY_N_DS	Input	Low	**Destination Ready**: Indicates that the destination interface is ready to accept data presented to it on the data bus in the current cycle. The DST_RDY _N_DS signal is passed through to the upstream interface without any latency.
RX_SRC_RDY_N_DS	Output	Low	**Source Ready**: Indicates data that is provided by the source interface on the data bus is valid during the current cycle. The SRC_RDY_N_US is passed through to the downstream interface with a fixed latency.
RX_REM_DS[p:0]	Output	N/A	**Remainder**: Indicates the byte offset of the last valid byte on the data bus for the last PDU transfer. Remainder value is valid concurrent end-of-frame assertions. The REM_US is appropriately modified to indicate the new value of REM if the user chooses the strip CRC option.

Table 6: **RXCRC Module Interface Signals *(Continued)***

Name	Direction	Active	Definition
RX_CRCPASS_FAIL_N	O	High	**Pass/Fail**: Indicates whether the incoming frame passes or fails the CRC verification. This signal is valid when CRC_VALID is asserted high. • A High logic level indicates CRC pass. • A Low logic level indicates CRC failure.
CRC_VALID	O	High	**Valid**: The RX_CRCPASS_FAIL_N is valid when CRC_VALID is asserted high.

Data Transfer Through RXCRC Module

Figure 5 shows the data transfer through the RXCRC block when the strip CRC option is chosen. The calculation of the CRC starts on the assertion of the SOF_N_US signal on the upstream interface.

There is fixed latency between the upstream and downstream interface. The value of this latency depends on the CRC chosen and the data width. The EOF_N_US signal terminates the calculation of the CRC. CRC is not calculated on any data after the EOF_N_US signal. In Figure 5, the CRC polynomial is assumed to be CRC16. As shown in Figure 5, the REM_US value coinciding with EOS_N_US is 3. This means that all four bytes of data are valid. When The CRC bytes are stripped, the REM value coinciding with EOF_N_DS is modified to 1, indicating that two bytes of data are valid.

The user should note that the CRC is stripped by realigning the EOF_N_DS signal and modifying the REM_DS. The CRC bytes are still on the data bus and should be ignored. This is especially important if the user application considers that valid data could occur even after the end-of-frame.

Figure 5 shows a packet that has passed the CRC verification. This is indicated by the CRCPASS_FAIL_N signal, which is high at CRC_VALID.

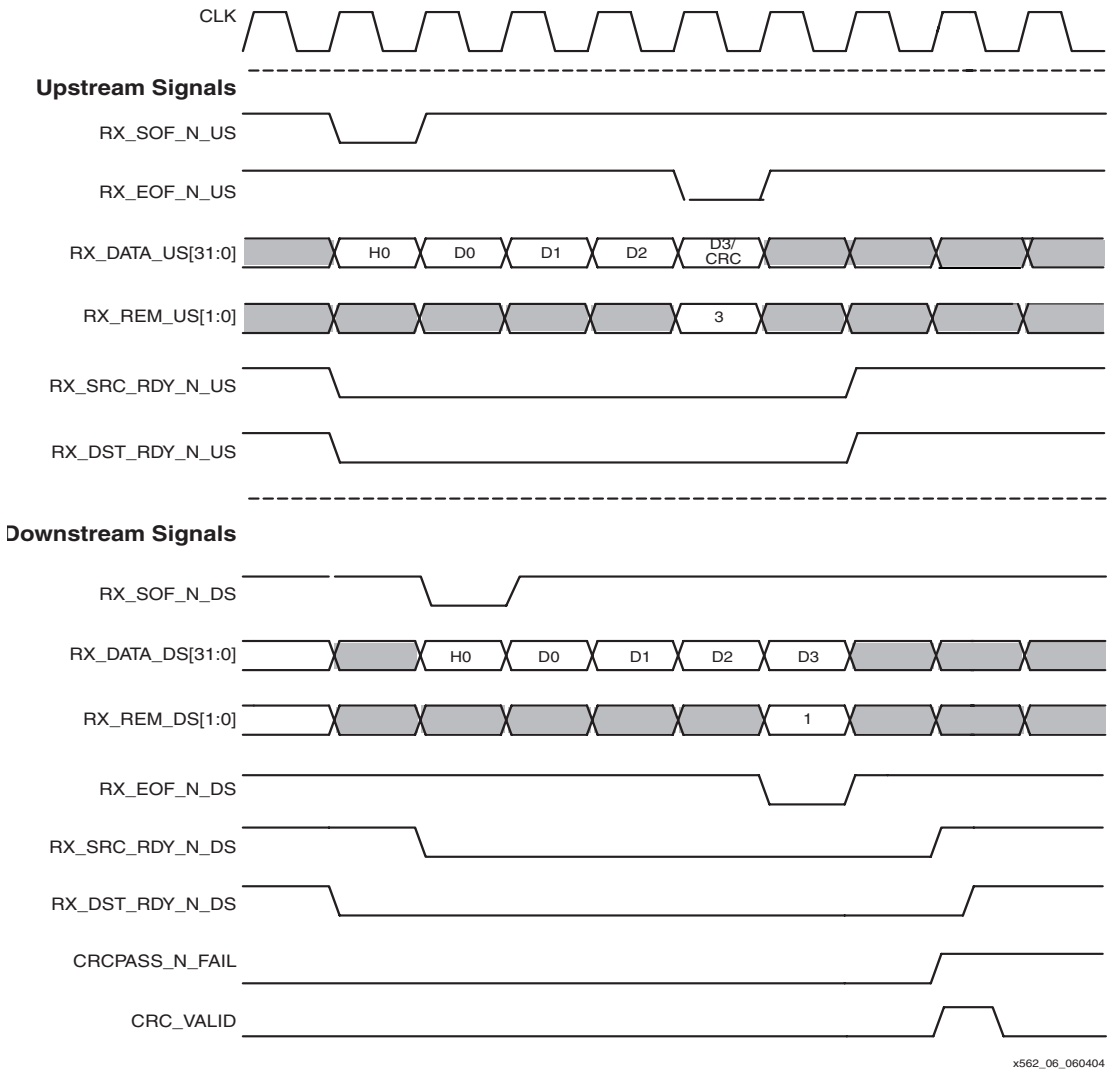

Figure 5: **Frame Transfers Through RXCRC Module - CRC Stripped**

Figure 6 shows the data transfer through the RXCRC block when the strip CRC option is not chosen. The CRC calculation starts on the assertion of the SOF_N_US signal on the upstream interface.

The EOF_N_US signal terminates the calculation of the CRC. In Figure 6, the CRC polynomial is assumed to be CRC16. Since the CRC is not stripped, the incoming frame is just passed on to the downstream interface without any modification. The additional signal CRC_PASS_N_FAIL is used to indicate whether the incoming packet passes the CRC verification. This signal is valid at CRC_VALID.

Figure 6 shows a frame that has passed the CRC verification. This is indicated by the CRCPASS_FAIL_N signal, which is high at CRC_VALID.

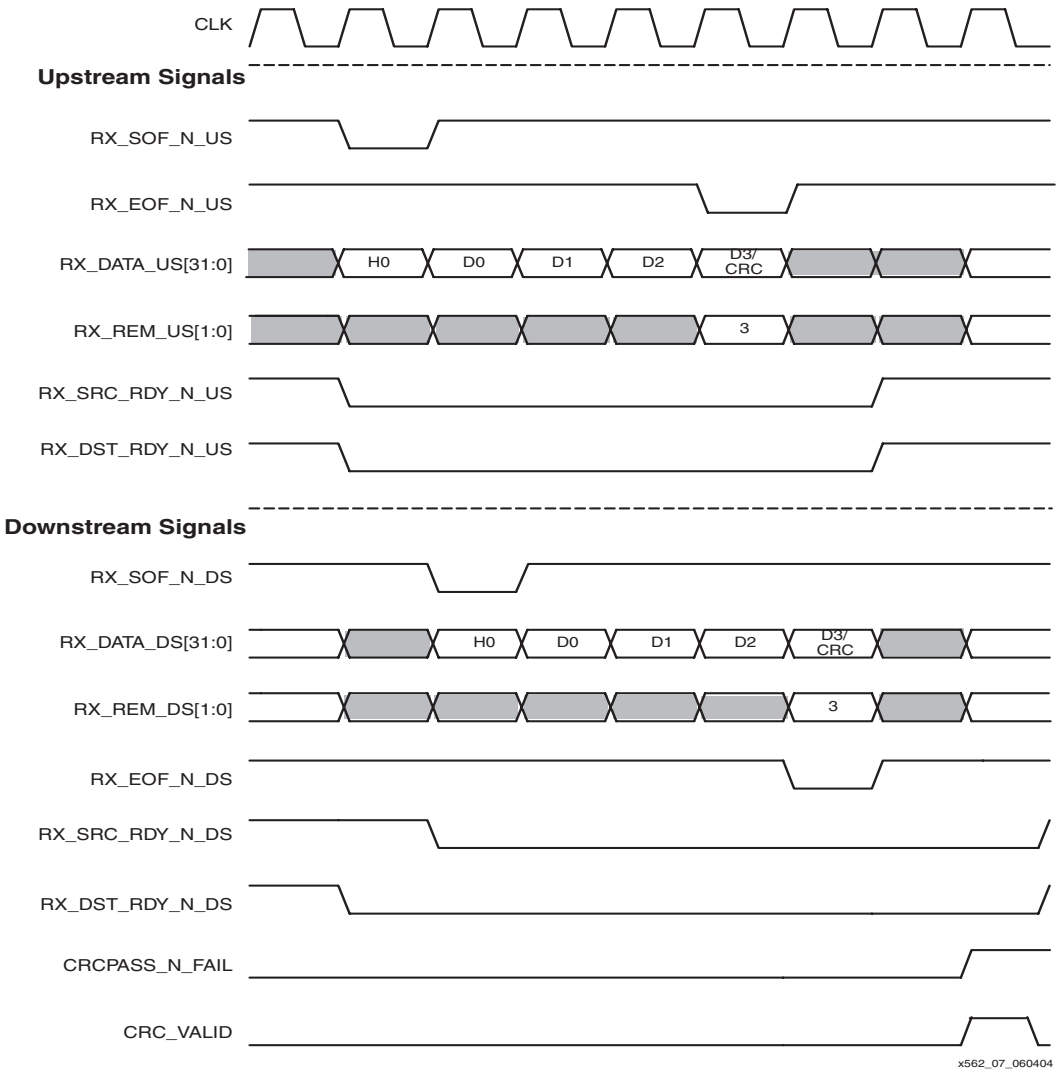

Figure 6: **Frame Transfers Through RXCRC Module - CRC Not Stripped**

LocalLink CRC Parameters

Table 7 provides descriptions of the LocalLink CRC parameters.

Table 7: **LocalLink CRC Parameters**

	Parameter Name	Legal Values	Default Value	Variable Type	Description
1	data_width	8 – 128 in multiples of 8	32	Integer	**Data Width**
2	crc_width	8, 16, 32, 64			**CRC Width**
3	poly		4C11DB7(hex)	String	**CRC Polynomial**: The user can specify any CRC polynomial. However, the maximum degree of the polynomial allowed is 64. Also, the polynomial value specified must have a coefficient of 1 in the LSB position and in the MSB position. The default value used is 4C11DB7(hex) which is the polynomial for CRC32.
4	Insertcrc	0 – deassert ST_RDY_N_US 1 – pad byte(1)	0	String	**Tx - Pad or DST_RDY_N_US Option**(2): This option allows the user to choose how the CRC bytes are inserted into the data stream in the TX block. The insertion can be done in two ways: • User provides pad bytes that are overwritten with the CRC. • CRC is appended while DST_RDY_N_US is deasserted.
5	aligned_data	0 – not aligned 1 – aligned	0	Bit	**Aligned Data**: This option restricts alignment of data. If this option is chosen, it results in a much smaller and faster module, especially in the non-pipelined case. Refer to Table 4 for resource numbers and speeds. When this option is chosen, the REM value is ignored and is always assumed to be all 1s.
6	pipeline	0 – not pipelined 1 – pipeline	0	Bit	**Pipeline Option**: This option allows the user to use a pipelined version of the CRC modules. In this case, the CRC calculation is pipelined by one stage, providing higher speeds. Refer to Table 4 for resource numbers and speeds.

Table 7: **LocalLink CRC Parameters *(Continued)***

	Parameter Name	Legal Values	Default Value	Variable Type	Description
7	crcstrip	0 – not stripped 1 – CRC stripped	0	Bit	**CRC Strip**: The CRC is stripped by moving RX_EOF_N_US to indicate the new End-of-Frame without the CRC Word(s) and also by modifying the RX_REM_US value on the downstream interface. The CRC Word(s) are still present in the frame.
8	back_to_back	0 – not back-to-back 1 – back-to-back	0	Bit	**Back-to-Back Data Transfers**: This option is available for the RXCRC module only. This option allows the RXCRC module to accept back-to-back frame transfers. This option results in a larger number of FFs for the design. If this option is not chosen, then the DST_RDY_N_US is deasserted for 1-2 cycles at the end of the frame.
9	residue_value		C704DD7B	String	**Residue Value**: This parameter is used to accept the residue value of the polynomial from the user. By default, this value is set to C704DD7B, which is the residue value for the standard 32-bit Ethernet polynomial.
10	crcinit		All ‘1’s	String	**Initial Value of CRC**: Most protocols specify a starting value of CRC of all 1s. This is the default starting value. The user can specify the desired starting value. The CRC register is loaded with the specified value at RESET.
11	compdata	0 – not complemented 1 – complemented	1	Bit	**Complement Input Data**: Some protocols specify that each data byte should be complemented before it is passed through the CRC calculator. This parameter allows the user to implement this feature.

Table 7: **LocalLink CRC Parameters *(Continued)***

	Parameter Name	Legal Values	Default Value	Variable Type	Description
12	trandata	0 – not transposed 1 – transposed	1	Bit	**Transpose Input Data Bytes**: If this option is set, each RX/TX_DATA_US byte is transposed before it is passed through the CRC equations. Transpose Input Data: For example, consider an input word of C0010203 (hex) for a data width of 32. Each byte in this word is transposed before it is passed through the CRC generator, i.e., the input to the CRC equations is 038040C0 (hex).
13	compCRC	0 – not transposed 1 – transposed	1	Bit	**Complement CRC Bytes**: This parameter is useful for protocols, such as Ethernet and Fibre Channel which require a final complementing of each of the CRC bytes.
14	tranCRC	0 – not transposed 1 – transposed	1	Bit	**Transpose CRC Bytes**: If this option is set, each byte of the CRC generated from the CRC generator is transposed. For example, consider a generated CRC of 9B61F408 (hex). Each byte is transposed to get D9862F10 (hex).

Notes:

1. This feature is not implemented in the current release of the reference design. The source files contain the insertcrc parameter, but a value of 1 for this parameter is invalid.
2. Only the DST_RDY_N_US is supported in the current release.

Implementation Block Diagrams

Figure 7 shows a simplified view of the implementation of the non-pipelined version of the TXCRC modules. The data from the upstream interface is passed through SRL16s to introduce the appropriate latency. The output of the SRL16 is then complemented and/or transposed according to the parameter specified by the user. This is then passed to the combinatorial CRC generator function. The function calculates the CRC only over the valid data as indicated by the REM_US value.

The incoming data is streamed to the output interface. The registered CRC is again complemented and/or transposed depending on the parameters. The CRC Word(s) is then inserted into the frame on the downstream interface.

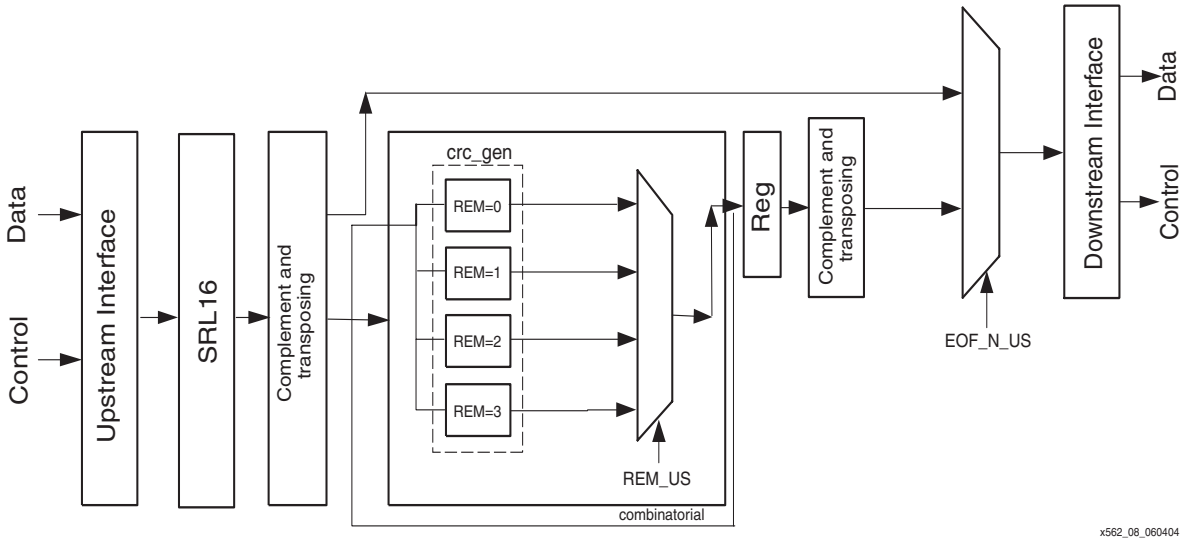

Figure 7: **Implementation Block Diagram of Non-Pipelined TXCRC Module**

Figure 8 shows the block diagram of the implementation for the pipelined version of the modules. As can be seen in Figure 8, the CRC is calculated for all possible values of REM and then registered to provide one stage of pipelining. These CRC values are then passed through a MUX to obtain the correct value of the CRC based on the input value of REM_US.

Figure 7 and Figure 8 are simplified drawings to illustrate the implementation. Different parameter values for different cases could cause some deviation. The RXCRC module is similar, with added logic for validation.

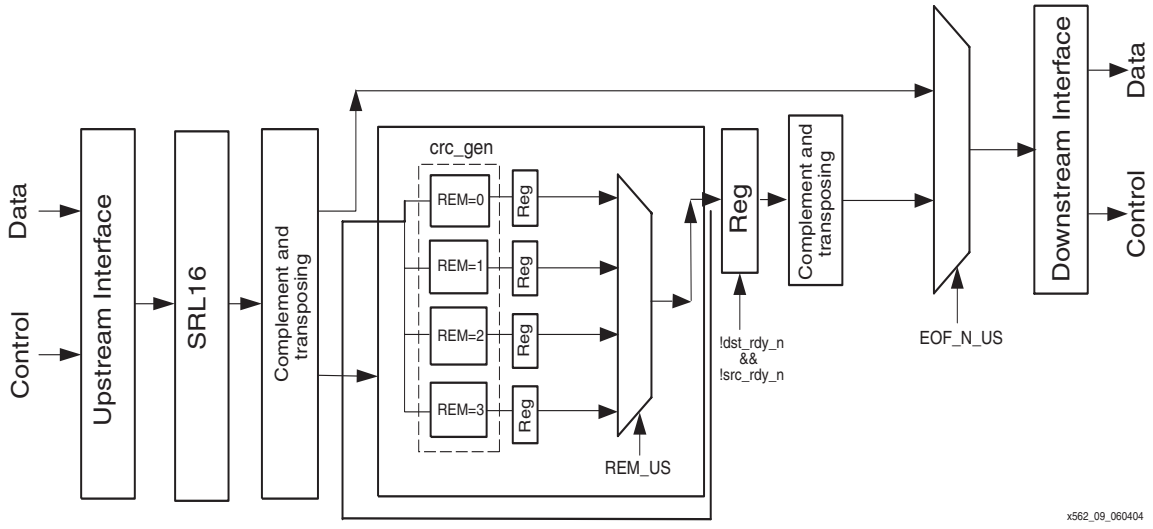

Figure 8: **Implementation Block Diagram of Pipelined RXCRC Modules**

Reference Design Files

The Configurable LocalLink CRC reference design contains full VHDL source code for the Transmit and Receive CRC modules described in this application note. The current release (v1.1) contains only VHDL source which is available in a downloadable ZIP file at: xapp562.zip.

Directory Structure

The directory structure for the reference design is illustrated in Figure 9.

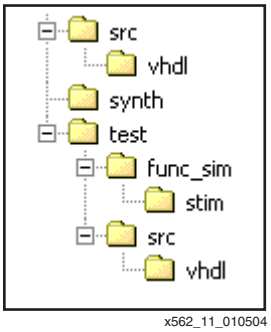

Figure 9: **Directory Structure**

The `src/` directory contains the implementation source code.

The `synth/` directory contains synthesis scripts for Synplify and XST.

The `test/` directory contains testbench source and run scripts for functional simulation.

Functional Simulation

Figure 10 shows the block diagram of the simulation testbench shipped with the reference design.

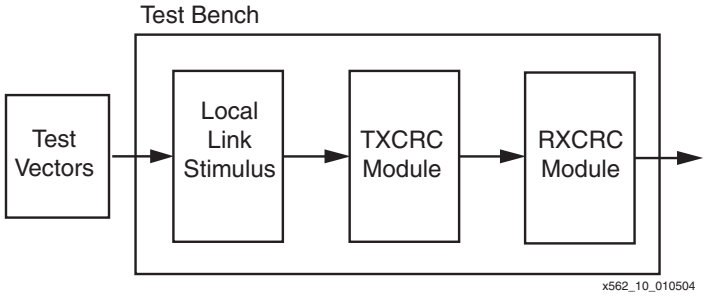

Figure 10: **Testbench Block Diagram**

The testbench instantiates both the TXCRC and RXCRC modules, setup so that the TXCRC module passes a frame with a generated CRC, and the RXCRC takes it as input and verifies the CRC, asserting CRCPASS_FAIL_N appropriately.

The parameters were chosen to match the CRC characteristics of Ethernet. This makes it easy to compare against existing captured frames from a real Ethernet network. Ten Ethernet frames captured from a live network are included in the `xapp562/func_sim/stim/` directory. These frames are read from the stimulus files by the LocalLink Stimulus module and used as stimulus in the testbench, which is shown in Figure 10.

References

- Xilinx *SP006: LocalLink Interface Specification*.
- Rajesh Nair, Gerry Ryan and Farivar Farzaneh, *A Symbol Based Algorithm for Hardware Implementation of Cyclic Redundancy Check,* Bay Networks.
- http://www.nobugconsulting.com/crc_details.pdf

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
01/22/04	1.0	Initial Xilinx Release.
11/02/04	1.1	Updated version.
04/20/07	1.1.1	Fixed link to Xilinx LocalLink lounge.