XAPP267 (v1.0) January 15, 2001

Parity Generation and Validation in Virtex-II Devices

Author: Lakshmi Gopalakrishnan

Summary

In data transmission systems the transmission channel itself is a source of data error. Hence the need to determine the validity of transmitted and received data. Parity generation and validation is a scheme to provide single bit error detection capabilites. This application note describes how to generate and validate parity in a design using the Virtex™-II architectural features including block RAM.

Introduction

The validity of data is essential in applications where there data is transmitted over long distances. Invalid data is a corruption risk. Parity generation helps in checking the validity of the data. This application note explains how parity is efficiently generated using the block RAM feature available in Virtex-II devices. Logic resource utilization is minimized since the data busses in the block RAM store the parity bit along with the data. This application note specifically covers 8-bit, 16-bit, and 32-bit parity checks.

The design detailed in this document has two modules, the parity generation block and the parity validation block.

The block RAM input and output data busses are represented by two busses for 9-bit width (8+1), 18-bit width (16+2), and 36-bit width (32+4) configurations. The ninth bit associated with each byte can store parity or error correction bits. No specific function is performed on this bit.

The separate bus for parity bits facilitates some designs. However, other designs safely use a 9-bit, 18-bit, or 36-bit bus by merging the regular data bus with the parity bus. Read/write and storage operations are identical for all bits, including the parity bits.

Parity Generation Block

The parity generation block generates the parity value from the input data. Figure 1 is a block diagram of the parity generation block. An n-bit parity generation block generates a parity bit for every n-bits of data. The number of bits taken into consideration for generating parity depends upon the kind of parity check desired. Parity checks are usually done on every 8-bits, 16-bits, or 32-bits of data, depending on the chosen block RAM configuration. Parity generation is done for every 8-bits using a chain of XOR gates. The parity value is then stored in the DIP bus along with the data input in the block RAM sharing the same address. In a block RAM the regular data-in bus (DI) and the parity data-in bus (when available) have a total width equal to the port width. For example the 36-bit port data width is represented by DI<31:0> and DIP<3:0>. Table 1 shows the port definitions for Figure 1.

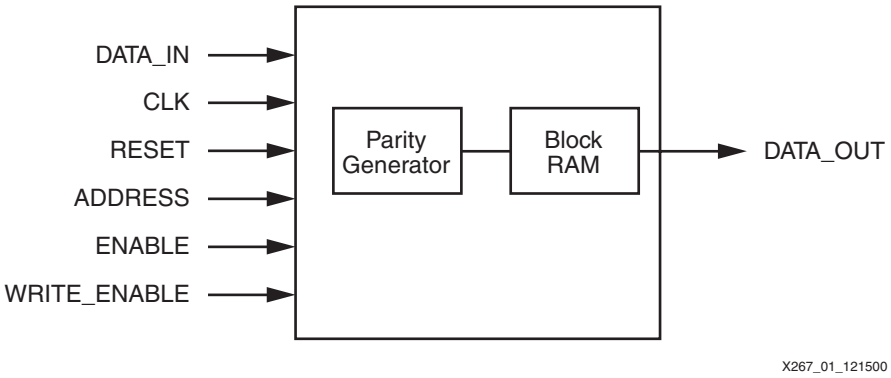

Figure 1: **Parity Generation Block Diagram**

Table 1: **Port Definitions**

Signal Name	Port Direction	Port Width
DATA_IN	Input	36
CLK	Input	1
RESET	Input	1
ADDRESS	Input	9
ENABLE	Input	1
WRITE_ENABLE	Input	1
DATA_OUT	Output	36

Parity Validation Block

The parity validation block validates the data that was received. Figure 2 illustrates the parity validation block diagram. It generates parity from the received data and compares it against the parity value that was stored in the block RAM by reading from the DOP bus. In the block RAM the regular data-out bus (DO) and the parity data-out bus (DOP) (when available) have a total width equal to the port width. The parity of the received data is generated using a chain of XORs. The parity is validated with a single XOR gate. A High on the Parity_Validity signal indicated a single bit error. The parity validity signal ensures the validity of the data received and helps to detect any single-bit errors. Table 2 gives port definitions for the parity validation block.

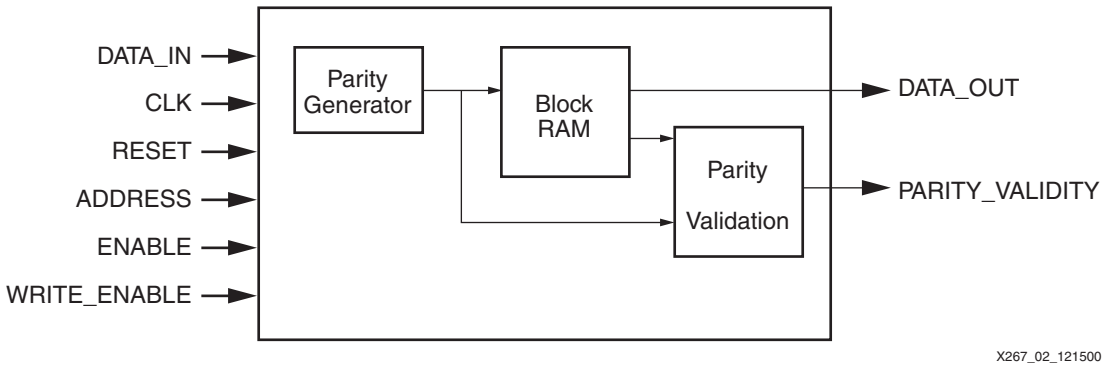

Figure 2: **Parity Validation Block Diagram**

Table 2: **Port Definitions**

Signal Name	Port Direction	Port Width
DATA_IN	Input	36
CLK	Input	1
RESET	Input	1
ADDRESS	Input	9
ENABLE	Input	1
WRITE_ENABLE	Input	1
PARITY_VALIDITY	Output	1
DATA_OUT	Output	32

Reference Design

The reference design **XAPP267.zip** is available for 8-bit, 16-bit, and 32-bit parity generation and checking modules in both VHDL and Verilog. The files have been tested and simulated using the ModelTech Simulator.

Table 3: **Reference Design Names and Descriptions**

Design	Description
Parity8_gen.vhd, .v	Parity generation for 8-bit data
Parity16_gen.vhd, .v	Parity generation for 16-bit data
Parity32_gen.vhd, .v	Parity generation for 32-bit data
Parity8_chk.vhd, .v	Parity check for 8-bit data
Parity16_chk.chd, .v	Parity check for 16-bit data
Parity32_chk.vhd, .v	Parity check for 32-bit data

Conclusion

Parity generation and validation can be performed efficiently using the block RAM available in Virtex-II devices. The separate bus for parity bits in the block RAM facilitate these designs.

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
01/15/01	1.0	Initial Xilinx release.