Lattice® Semiconductor Corporation

GAL® 20XV10: Data Block Transfer Address Detector

Introduction

The Exclusive-OR (XOR) gate can efficiently implement arithmetic functions such as counters, adders and decoders, using fewer product terms than the standard sum-of-products PLD architecture (AND, fixed OR array). This is demonstrated in Example 1.

To take full advantage of product term usage in a high-speed system design, a high-speed device with a built-in XOR function is needed. The GAL20XV10 fills the need for such a device. The GAL20XV10 achieves a 10ns Tpd while consuming only 90mA Icc (max.). The closest competitor's device offers only a Tpd of 30ns at 180mA Icc. In addition, the generic architecture of the GAL20XV10 gives system designers the ability to configure outputs to any combination of registers, combinatorial, XOR and AND/OR structures.

Design Example

An address counter that uses a comparator to keep track of the block data transfer is a typical application which illustrates the advantages of the GAL20XV10's XOR architecture. If the starting and ending addresses are given, the address counter will generate and increment the transfer address. The comparator will then compare the counter bits with the ending address. When the counter value equals the ending address, the address comparator will issue a transfer complete signal. The following CUPL example source file (Example 2) shows how this function can be implemented using CUPL compiler syntax. Notice that the syntax demonstrates the usage of .OE and .OEMUX to control the AND/OR product term configuration and XOR configuration, respectively.

Conclusion

This design example illustrates the efficient usage of the XOR function by implementing the address counter with 11 product terms instead of the 14 product terms required with a standard programmable AND, fixed OR configuration. The bit-wise comparator, implemented with the XOR function, also makes the design clear and understandable, as illustrated by the logic equations. With this efficient, easy to understand design, the system can run at up to 100MHz with 10ns tpd.

Technical Support Assistance

Hotline:	1-800-LATTICE (Domestic)
	1-408-826-6002 (International)
e-mail:	techsupport@latticesemi.com

Example 1. XOR Logic Equation

$ - XOR function syntax	& - AND function syntax
# - OR function syntax	! - INVERT function syntax

XOR Function / **Equivalent AND/OR Function**

```
 A $ B            /*  2 PT used    (A & !B) # (!A & B)           /* 2 PT used
(A & B) $ (C & D) /*  2 PT used    (A & B & !C) # (A & B & !D)   /* 4 PT used
                                   # (!A & C & D) # (!B & C & D)
```

Example 2. CUPL Source File

```
Name      APPXV10;
Partno    00;
Date      09/09/99;
Revision  00;
Designer  Jane Doe;
Company   Lattice;
Assembly  None;
Location  None;
Device    g20xv10;

/*******************************************************/
/*  This CUPL example uses the GAL20XV10 to build the */
/*  4-bit up counter with load function and a 4-bit   */
/*  comparator.  This  counter implementation takes   */
/*  advantage of the built-in XOR function of the     */
/*  GAL20XV10. It also shows the XOR and AND/OR       */
/*  configuration in CUPL syntax                      */
/*******************************************************/

/**  Input definition  **/

PIN 1       = SYSCLK;
PIN 2       = SA0;      /* STARTING ADDRESS BITS */
PIN 3       = SA1;
PIN 4       = SA2;
PIN 5       = SA3;
PIN 6       = EA0;      /* ENDING ADDRESS BITS */
PIN 7       = EA1;
PIN 8       = EA2;
PIN 9       = EA3;
PIN 10      = STARTLD;  /* STARTING ADDRESS LOAD */
PIN 11      = OE_COMP;
PIN 13      = OUT_EN;

/**  Output Definition  **/

PIN  23 = !AC0;         /* ADDRESS COUNTER BITS */
PIN  22 = !AC1;
PIN  21 = !AC2;
PIN  20 = !AC3;
PIN  19 = !CMP0;        /* ADDRESS COMPARE BITS */
PIN  18 = !CMP1;
PIN  17 = !CMP2;
PIN  16 = !CMP3;
PIN  15 =  EQUAL;       /* EQUALITY COMPARE */

/**  Equations  **/

AC0.D  =  !STARTLD & AC0            /** AC0 TOGGLE WITH CLOCK **/
        $  STARTLD & SA0;           /** LOAD SA0 **/
AC0.OEMUX  =  OUT_EN;
```

```
AC1.D  =  !STARTLD & AC0                  /** AC1 CNT UP CONDITION  **/
        $ !STARTLD & AC1                  /** TOGGLE AC1  **/
        #  STARTLD & SA1;                 /** LOAD SA1 **/
AC1.OEMUX  =  OUT_EN;

AC2.D  =  !STARTLD & AC0 & AC1            /** AC2 CNT UP CONDITION  **/
        $ !STARTLD & AC2                  /** TOGGLE AC2  **/
        #  STARTLD & SA2;                 /** LOAD SA2 **/
AC2.OEMUX  =  OUT_EN;

AC3.D  =  !STARTLD & AC0 & AC1 & AC2      /** AC3 CNT UP CONDITION  **/
        $ !STARTLD & AC3                  /** TOGGLE AC3  **/
        #  STARTLD & SA3;                 /** LOAD SA3 **/
AC3.OEMUX  =  OUT_EN;

CMP0 = AC0 $ EA0;                   /** COMPARE ADDR BIT0  **/
CMP0.OEMUX = OUT_EN;

CMP1 = AC1 $ EA1;                   /** COMPARE ADDR BIT1  **/
CMP1.OEMUX = OUT_EN;

CMP2 = AC2 $ EA2;                   /** COMPARE ADDR BIT2  **/
CMP2.OEMUX = OUT_EN;

CMP3 = AC3 $ EA3;                   /** COMPARE ADDR BIT3  **/
CMP3.OEMUX = OUT_EN;

EQUAL = !CMP0 & !CMP1 & !CMP2 & !CMP3;    /** MAGNITUDE COMPARE **/
EQUAL.OE = OE_COMP;
```