

Introduction

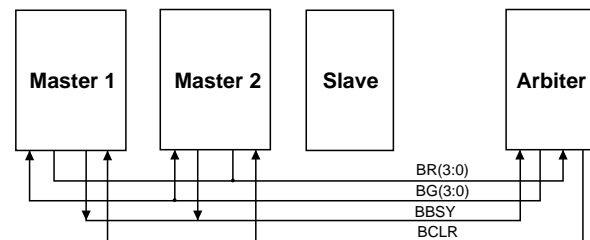
The GAL22V10 provides a quick solution to bus arbitration and control needs. In this application note, we discuss how a VME bus arbitration circuit can be easily implemented within a GAL22V10, while leaving logic available on the GAL for other functions.

In any bus-oriented system, there may be multiple devices that need exclusive access to the bus. You need to provide some form of arbitration, ensuring that only one device has control of the bus at any point in time. This arbitration function is an ideal candidate for a GAL-based solution. For this example, we have chosen the VME bus standard and describe an approach for a GAL22V10 VME bus arbiter, including the ABEL source code for programming.

Design Example

A VME-based system is a good example of a bus that supports multiple bus masters, by requiring bus arbitration logic on the bus to resolve possible conflicting requests. The VME bus has a bus request line that a device asserts when it wants to gain control of the bus. The arbiter prioritizes these bus requests and asserts a bus grant to the device with the highest priority. Since you may wish to give some devices higher priority to bus access than others, the VME bus provides four bus request lines, with BR3 designated as the highest priority request, and BR0 as the lowest priority request. And since you may have more than four devices on the bus, or you may want to have more than one device sharing the same priority level, the bus standard must also support multiple masters with the same priority level. The VME bus allows more than one device on the bus to share a bus request line by "daisy-chaining" the bus grant signal — the device physically closest to the bus arbitrator "sees" the bus grant first. If this device didn't generate the bus request, then the bus grant is allowed to proceed to the next device on the bus. Figure 1 shows a simplified block diagram of a VME system, showing the handshaking between the arbiter and the other devices on the bus.

Figure 1. The Bus Arbitration Process



The bus arbitration process is defined in the following steps (the *level* of a bus master indicates which bus request line it used to gain access to the bus):

- 1) A bus request signal (BR3:0) is received by the arbiter.
- 2) If the bus is not busy, the arbiter returns a corresponding bus grant (BG3:0)
- 3) If the bus is busy, the arbiter asserts a bus clear request (BCLR) as long as one of the following conditions is true:
 - The current bus master is level 2, 1 or 0, and the active bus request line is 3 or 2.
 - The current bus master is level 0, and any other device is requesting the bus

This protocol allows a higher priority device to take control of the bus in an orderly manner — the arbiter asserts BCLR, the current bus master releases BBSY, and then the higher priority device gains control of the bus. Also note that if the current bus master is 3, then the arbiter gives control to another level 3 requester only when BBSY goes inactive — in other words, a level 3 master will be allowed uninterrupted access to the bus.

ABEL Source

The following is an ABEL source file that implements a VME bus arbiter in a GAL22V10. Note that the internal signals, Master1 and Master2, are used to hold the level of the bus request currently being served.

VME Bus Arbitration Using a GAL22V10

Example 1: Module Arbiter

Title

Bus Arbiter with Priority Handling for GAL22V10
Lattice Semiconductor'

Declarations

```
    arbiter device 'p22v10';
```

"Inputs

```
    CLK                pin 1;
    !BBUSY              pin 2;
    !BR0,!BR1,!BR2,!BR3 pin 3,4,5,6;
    !RESET              pin 11;
    !OE                 pin 13;
```

"Outputs

```
    !BCLR               pin 23  istype 'invert,reg_d';
    !BGIN0,!BGIN1,!BGIN2,!BGIN3 pin 22,21,20,19 istype 'invert,reg_d';
    MASTER1,MASTER0     pin 15,14 istype 'buffer,reg_d';
```

```
    MASTER = [MASTER1, MASTER0]; C, X, Z, L, H = .C., .X., .Z., 0, 1;
```

Equations

```
    BGIN3      := !BBUSY & BR3
               # !BBUSY & BGIN3;
    BGIN3.C    = CLK;
    BGIN3.AR   = RESET;
    BGIN3.OE   = OE;

    BGIN2      := !BBUSY & !BR3 & BR2
               # !BBUSY & BGIN2;
    BGIN2.C    = CLK;
    BGIN2.AR   = RESET;
    BGIN2.OE   = OE;

    BGIN1      := !BBUSY & !BR3 & !BR2 & BR1
               # !BBUSY & BGIN1;
    BGIN1.C    = CLK;
    BGIN1.AR   = RESET;
    BGIN1.OE   = OE;
    BGIN0      := !BBUSY & !BR3 & !BR2 & !BR1 & BR0
               # !BBUSY & BGIN0;
    BGIN0.C    = CLK;
    BGIN0.AR   = RESET;
    BGIN0.OE   = OE;

    MASTER1    := BBUSY & BGIN3      "Master MSB
               # BBUSY & BGIN2
               # BBUSY & MASTER1;
    MASTER0    := BBUSY & BGIN3      "Master LSB
               # BBUSY & BGIN1
               # BBUSY & MASTER0;
    MASTER.AR   = RESET;
    MASTER.OE   = OE;
    MASTER.C    = CLK;

    BCLR       := BBUSY & BR3 & (MASTER <= 2)
               # BBUSY & BR2 & (MASTER <= 2)
               # BBUSY & BR1 & (MASTER <= 1)
               # BBUSY & BR0 & (MASTER == 0)
               # BBUSY & BCLR;
    BCLR.AR    = RESET;
```

VME Bus Arbitration Using a GAL22V10

```
BCLR.OE      = OE;
BCLR.C       = CLK;
```

Test_vectors

```
([CLK,!RESET,!BBUSY,!BR3,!BR2,!BR1,!BR0,!OE]->
[!BCLR,!BGIN3,!BGIN2,!BGIN1,!BGIN0,MASTER])
```

```
"      ! !                ! ! ! ! M
"      R B                ! B B B B A
"      E B ! ! ! !      B G G G G S
" C S U B B B B !      C I I I I T
" L E S R R R R O      L N N N N E
" L T Y 3 2 1 0 E      R 3 2 1 0 R
"
[X,X,X,X,X,X,X,1]->[Z,Z,Z,Z,Z,Z];"tristate
[C,1,1,1,1,1,0,0]->[H,H,H,H,L,0];"BR0 request
[C,1,1,1,1,1,0,0]->[H,H,H,H,L,0];
[C,1,1,1,1,1,0,0]->[H,H,H,H,L,0];
[C,1,0,1,1,1,1,0]->[H,H,H,H,H,0];
[C,1,0,1,1,1,1,0]->[H,H,H,H,H,0];
[C,1,0,1,1,1,0,0]->[L,H,H,H,H,0];"test bus clear line >= 0
[C,1,1,1,1,0,0,0]->[H,H,H,L,H,0];"BR1 request higher priority
[C,1,0,1,1,1,1,0]->[H,H,H,H,H,1];
[C,1,0,1,1,1,0,0]->[H,H,H,H,H,1];"test bus clear line >= 1
[C,1,0,1,1,0,1,0]->[L,H,H,H,H,1];

[C,1,1,1,0,0,0,0]->[H,H,L,H,H,0];                "BR2 request higher priority
[C,1,0,1,1,1,1,0]->[H,H,H,H,H,2];
[C,1,0,1,1,1,0,0]->[H,H,H,H,H,2];                "test bus clear line >= 2
[C,1,0,1,1,0,1,0]->[H,H,H,H,H,2];
[C,1,0,1,0,1,1,0]->[L,H,H,H,H,2];

[C,1,1,0,1,1,1,0]->[H,L,H,H,H,0];                "BR3 request higher priority
[C,1,0,1,1,1,1,0]->[H,H,H,H,H,3];
[C,1,0,1,1,1,0,0]->[H,H,H,H,H,3];                "test bus clear line
[C,1,0,1,1,0,1,0]->[H,H,H,H,H,3];                "requests ignored
[C,1,0,1,0,1,1,0]->[H,H,H,H,H,3];
[C,1,0,0,1,1,1,0]->[H,H,H,H,H,3];
[X,0,X,X,X,X,X,0]->[H,H,H,H,H,0];                "reset
```

END



Copyright © 1996 Lattice Semiconductor Corporation.

E²CMOS, GAL, ispGAL, ispLSI, pLSI, pDS, Silicon Forest, UltraMOS, Lattice Logo, L with Lattice Semiconductor Corp. and L (Stylized) are registered trademarks of Lattice Semiconductor Corporation (LSC). The LSC Logo, Generic Array Logic, In-System Programmability, In-System Programmable, ISP, ispATE, ispCODE, ispDOWNLOAD, ispGDS, ispStarter, ispSTREAM, ispTEST, ispTURBO, Latch-Lock, pDS+, RFT, Total ISP and Twin GLB are trademarks of Lattice Semiconductor Corporation. ISP is a service mark of Lattice Semiconductor Corporation. All brand names or product names mentioned are trademarks or registered trademarks of their respective holders.

Lattice Semiconductor Corporation (LSC) products are made under one or more of the following U.S. and international patents: 4,761,768 US, 4,766,569 US, 4,833,646 US, 4,852,044 US, 4,855,954 US, 4,879,688 US, 4,887,239 US, 4,896,296 US, 5,130,574 US, 5,138,198 US, 5,162,679 US, 5,191,243 US, 5,204,556 US, 5,231,315 US, 5,231,316 US, 5,237,218 US, 5,245,226 US, 5,251,169 US, 5,272,666 US, 5,281,906 US, 5,295,095 US, 5,329,179 US, 5,331,590 US, 5,336,951 US, 5,353,246 US, 5,357,156 US, 5,359,573 US, 5,394,033 US, 5,394,037 US, 5,404,055 US, 5,418,390 US, 5,493,205 US, 0194091 EP, 0196771B1 EP, 0267271 EP, 0196771 UK, 0194091 GB, 0196771 WG, P3686070.0-08 WG. LSC does not represent that products described herein are free from patent infringement or from any third-party right.

The specifications and information herein are subject to change without notice. Lattice Semiconductor Corporation (LSC) reserves the right to discontinue any product or service without notice and assumes no obligation to correct any errors contained herein or to advise any user of this document of any correction if such be made. LSC recommends its customers obtain the latest version of the relevant information to establish, before ordering, that the information being relied upon is current.

LSC warrants performance of its products to current and applicable specifications in accordance with LSC's standard warranty. Testing and other quality control procedures are performed to the extent LSC deems necessary. Specific testing of all parameters of each product is not necessarily performed, unless mandated by government requirements.

LSC assumes no liability for applications assistance, customer's product design, software performance, or infringements of patents or services arising from the use of the products and services described herein.

LSC products are not authorized for use in life-support applications, devices or systems. Inclusion of LSC products in such applications is prohibited.

LATTICE SEMICONDUCTOR CORPORATION

5555 Northeast Moore Court
Hillsboro, Oregon 97124 U.S.A.

Tel.: (503) 681-0118

FAX: (503) 681-3037

<http://www.latticesemi.com>

November 1996
