Technical Note

TN227

Interfacing External I/O with Rabbit 2000/3000 Designs

Introduction

This technical note covers suggestions for interfacing I/O devices to the Rabbit 2000 and 3000 microprocessors. In particular, software provisions are described to handle devices that require a longer address hold time than is provided by the Rabbit microprocessor.

Background

As shown in Figure 1 below, an I/O cycle is 3 clocks long (or longer if wait states are added). Generally the data and address hold times for the write cycle are at least 10 ns. The exact amount depends on the clock speed and the setup of the clock doubler. However, the address hold time for the read cycle is quite small, nominally zero, but is usually positive because of capacitive loading on the address lines, which delays the address lines relative to the chip select and the read strobe. If the read strobe or chip select are delayed by either capacitive loading or additional glue logic, the address hold time can become negative. The read strobe is not usually heavily loaded, so the address hold time remains positive.

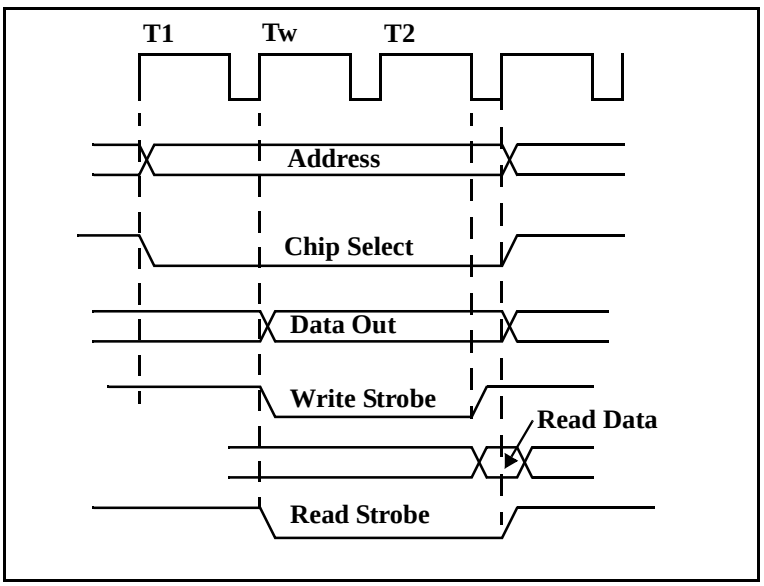

Figure 1. I/O Cycle Timing (no extra wait states)

This behavior of the chip select, read strobe, and address hold time applies to the main-memory I/O bus. If the auxiliary I/O bus, which is available only on the Rabbit 3000, is used, then the address and data out are held until the next I/O cycle. There is a minimum separation of one clock between I/O bus cycles, so the clock address and data hold time are at least one clock cycle long.

Several approaches are available to increase the address hold time when interfacing I/O devices to the main bus.

- If only a few nanoseconds are required, a series resistor can be placed between the address bus and the device. This creates an RC delay, with the capacitance supplied by the input capacitance of the device. Typically 500 or 1000 Ω in each line will assure 5 ns of hold time. Do *not* place the resistors between the Rabbit microprocessor and the memories or other devices that cannot tolerate additional address hold-time delays.
- Another approach is to not use the address lines at all, but use parallel outputs to serve as address lines. In this case, the hold time can be as long as desired. Generally a read strobe from Parallel Port E should be used to ensure that a double-select of two devices cannot occur. If the /IORD signal is used for several devices, a double select could occur if the chip select for one device is enabled and then another device is read while the chip select for the first device is still enabled.
- If a longer address hold time is required, a transparent latch with a delayed strobe can be used. In this case it is important not to hold the address for such a long time that it is not correctly updated before the next write strobe. Such an error could result in writing to the wrong address (but only in level-triggered devices, such as memories). The fastest that a write strobe can follow the end of a read strobe is about 6 clocks, for example, with the instructions **`IOE LD A,(HL)`** followed by **`IOE LD (HL),B`**. A more likely error is that the transparent latch would not freeze its output before the end of the cycle, but this can be prevented by using wait states. If 7 wait states are selected, the enable delay for the transparent latch can be as long as about 5 clocks without bad effects. This is at least 100 ns even on a Rabbit 3000 running at 50 MHz. Figure 2 shows a circuit that should provide at least 25 ns of address hold time, but never as much as 100 ns.

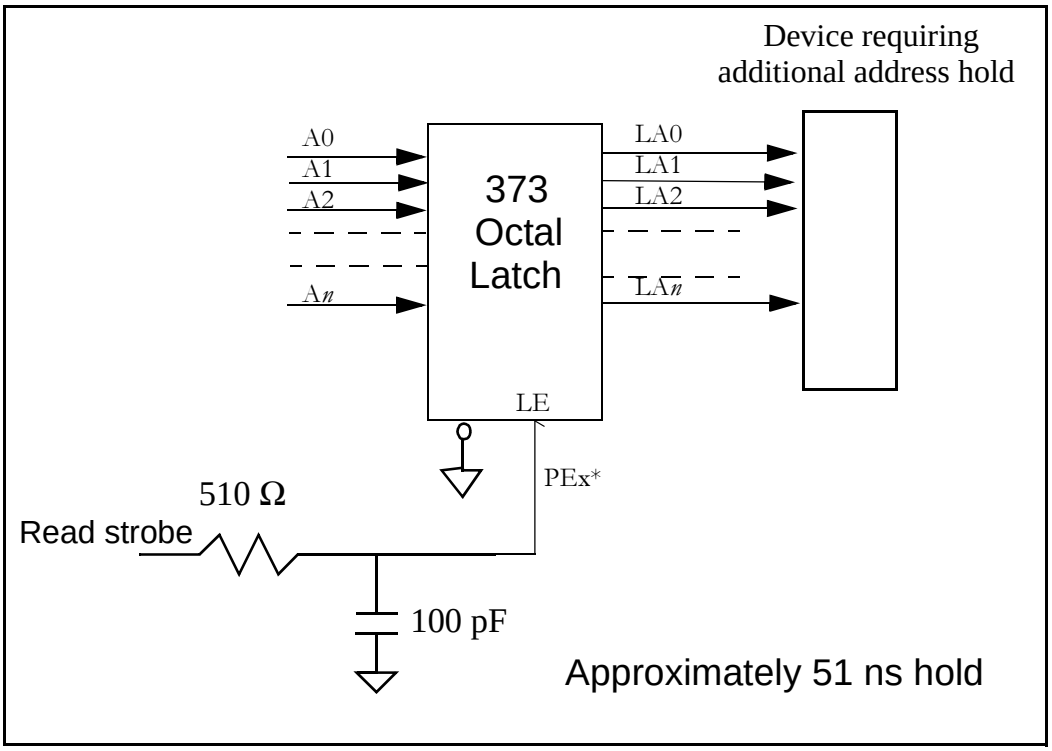

Figure 2. Address Hold Using a Transparent Latch

Software Alignment

It is possible to get additional hold time with software. The technique is most useful if only a small (3 or 4) number of address lines are being used. Align the I/O instruction such that the next memory cycle following the I/O bus cycle will have the same value for the address lines in question. This works if the I/O instruction is followed by the instruction fetch of the next instruction and that instruction is aligned so that the address lines are held as desired. This involves aligning the I/O instruction at a certain memory address. It is also necessary to disable the interrupts to ensure that no interrupt takes place between the I/O instruction and the following instruction fetch.

This technique will provide at least 2 CPU clock cycles of hold time for those address lines because the address lines won't change state after the I/O access. It is not necessary to match all the address lines—only the ones used in decoding the I/O address are required. If A0–A3 are matched, the software needs to align the access routine on a 16-byte boundary—the memory requirements increase if you try to match additional address lines. Decoding 4 address lines gives 16 addresses for the I/O device being addressed—this should be adequate for most purposes. The address hold time will be longer if you have inserted wait states for your memory device where the aligned software routine resides (RAM vs. flash).

Let's look at an example for a CPU running at 22.1 MHz.

- Memory has no wait states:

  Address hold-time = processor CLK period × 2 CPU CLK cycles
  = 45.2 ns × 2 = 90.4 ns

- Memory with 1 wait state:

  Address hold-time = processor CLK period × (2 CPU CLK cycles + number of wait states)
  = 45.2 ns × (2 + 1 ws) = 135.6 ns

In the timing diagram below, the address lines are shown as being low in the active state. The function automatically aligns the function address to any of the 16 possible states of the lower addresses to provide the necessary address hold:

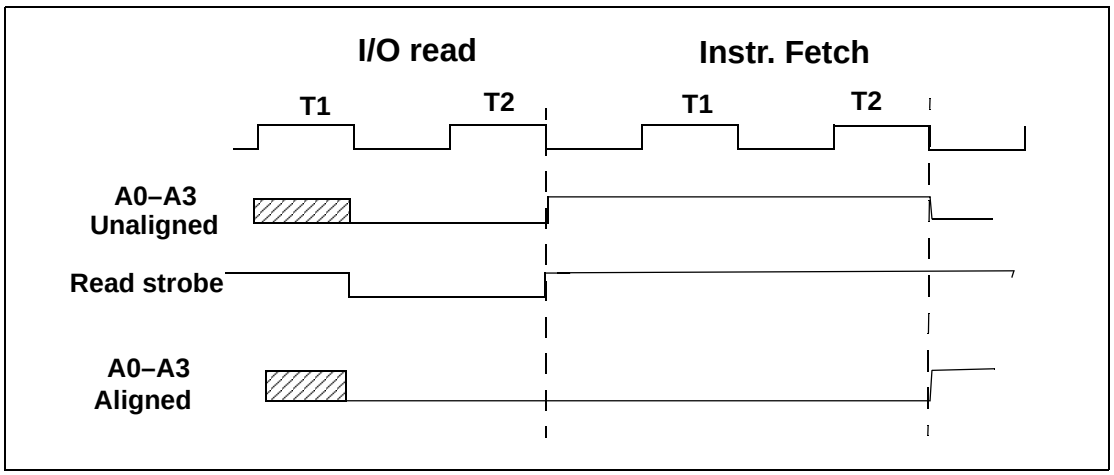

Figure 3. Timing Diagram for Aligned and Unaligned Address Lines

Software Implementation

To allow an additional margin for many hardware designs, this software technique has been implemented in the **IOE_RD** library that you can download from Z-World's Bulletin Board. The **RdExtPort** function in the library can be used to access the entire external I/O address range of 0x0000–0xFFFF, which will automatically provide two CPU CLK cycles of address hold-time for address lines A3–A0.

These steps provide detailed instructions for accessing and using this software technique.

1. Get **IOE_RD.zip** from Z-World's Bulletin Board. The ZIP file contains two files:

   IOE_RD.LIB

   IOE_RD_EXAMPLE.c

2. Copy the **IOE_RD.LIB** file to the Dynamic C **Lib** directory.
3. Add the line **\LIB\IOE_RD.LIB** to the **LIB.DIR** file.
4. Add the line **#use IOE_RD.lib** to your application program.
5. Initialize your I/O strobe(s).
6. Use the **RdExtPort(int port_address)** function call for external I/O read accesses that need the additional address hold-time.

The **IOE_RD_EXAMPLE.c** sample program illustrates how the **RdExtPort(int port_address)** function call is used.

The **IOE_RD.LIB** file is already included with Dynamic C versions 7.30 and above, and so does not need to be added to the **LIB.DIR** file as described in steps 1–6.

Z-World, Inc.

2900 Spafford Street
Davis, California 95616-6800
USA

Telephone: (530) 757-3737
Fax: (530) 757-3792

www.zworld.com

Rabbit Semiconductor

2932 Spafford Street
Davis, California 95616-6800
USA

Telephone: (530) 757-8400
Fax: (530) 757-8402

www.rabbitsemiconductor.com