DUAL-PORT SRAMs SIMPLIFY COMMUNICATION IN COMPUTER SYSTEMS

APPLICATION NOTE AN-02

By David C. Wyland

Introduction

Dual-port SRAMs allow two independent devices to have simultaneous access to the same memory. This allows the two devices to communicate with each other by passing data through the common memory. These devices might be a CPU and a disc controller or two CPUs working on different but related tasks. The dual-port memory approach is useful and popular because it allows the same memory to be used for both working storage and communication by both devices and avoids the need for any special data communication hardware between the devices. The latest development in dual-port SRAMs has been the appearance of high speed devices. These devices allow high speed access by both devices with the minimum amount of interference and delay.

Dual-Port RAMS: Simultanous Access

A dual-port memory has two sets of address, data and read/write control signals, each of which access the same set of memory cells. This is shown in Figure 1. Each set of memory controls can independently and simultaneously read any word in the memory including the case where both sides are accessing the same memory location at the same time. Standard memories have a single set of controls for address, data and read/write logic and are single-port RAMs. If you want a dual-port SRAM function, you need to design special logic to make the single-port RAM simulate a dual-port RAM in operation.

Direct Memory Access (DMA) as a Dual-Port Memory Simulation

The concept of using a conventional memory to simulate a dual-port RAM has been common in computer systems almost from the beginning. It is known under the name Direct Memory Access, or DMA. In the DMA concept, a single memory is shared between the CPU and one or more I/O devices as shown in Figure 2.

Each device wishing to use memory submits a request to the arbitration logic. The arbitration logic responds by connecting the memory address, data and control lines to one of the requesters and tells any other requesting devices to wait by issuing a $\overline{BUSY}$ signal. The $\overline{BUSY}$ signal causes the memory access logic in the device to wait until $\overline{BUSY}$ has gone away before performing a memory transfer.

DMA Limitations: Waiting for the Bus

In a computer system with DMA, the CPU must stop and wait while an I/O device is doing DMA transfers to memory. This works well in typical systems where the I/O devices are transferring data only a small percentage of the time and the impact on CPU processing time is minimal. These assumptions do not hold where you have two CPUs trying to use the same memory. In this case, one CPU must wait while the other uses the memory. As a result, the average speed of the CPUs will typically be cut in half.

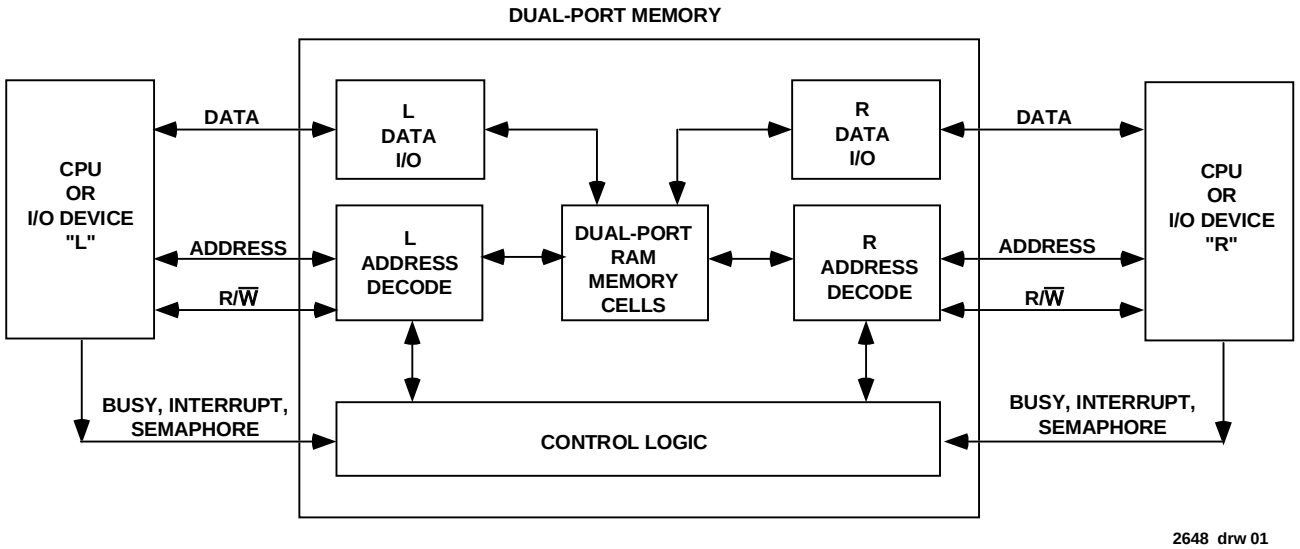


Figure 1. Dual-Port Memory Block Diagram

MARCH 1999

There are two solutions to this problem: 1) You can provide local memory for both CPUs and limit use of the common memory to CPU/CPU communication only in an attempt to reduce the time impact of DMA waiting, or 2) you can provide true hardware dual-port memory between the CPUs and all simultaneous high-speed access by both CPUs to the same memory without waiting. The introduction of high-speed dual-port SRAM chips now makes the second option practical.

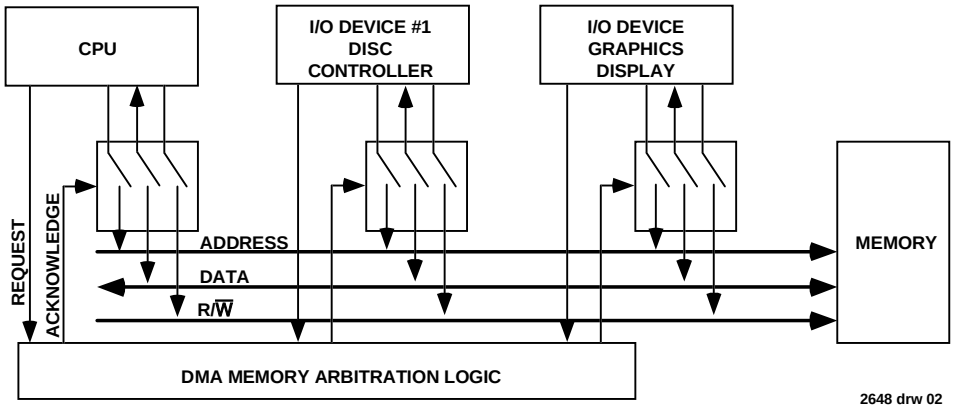


Figure 2. DMA Memory System Block Diagram

Dual-Port SRAM Chips: How They Work

A true dual-port memory allows independent and simultaneous access of the same memory cells by both devices. This means two complete and independent sets of address, data and read/write logic and memory cells that are capable of being read and written to by two different sources. An example of the dual-port memory cell is shown in Figure 3. In this cell both the left and right hand select lines can independently and simultaneously select the cell for read out. In addition, either side can write data into the cell independent of the other side. The only problem would be when both sides try to write into the same cell at the same time or simultaneous read/write. We will discuss this in a moment.

Dual-Port SRAM Control Logic

Dual-port SRAMs include control logic to solve three common application problems: signaling between processors, timing interactions when both are using the same location and hardware support for temporary assignment (called allocation) of a block of memory to one side only.

Interrupt Logic For Signaling

A common problem in dual-processor systems is signaling between the processors. For example, processor A needs to signal processor B to request a task to be performed, as defined by data in the common memory. When processor B has completed the task, it needs to signal processor A that the task is done. Note that the signaling must occur in both

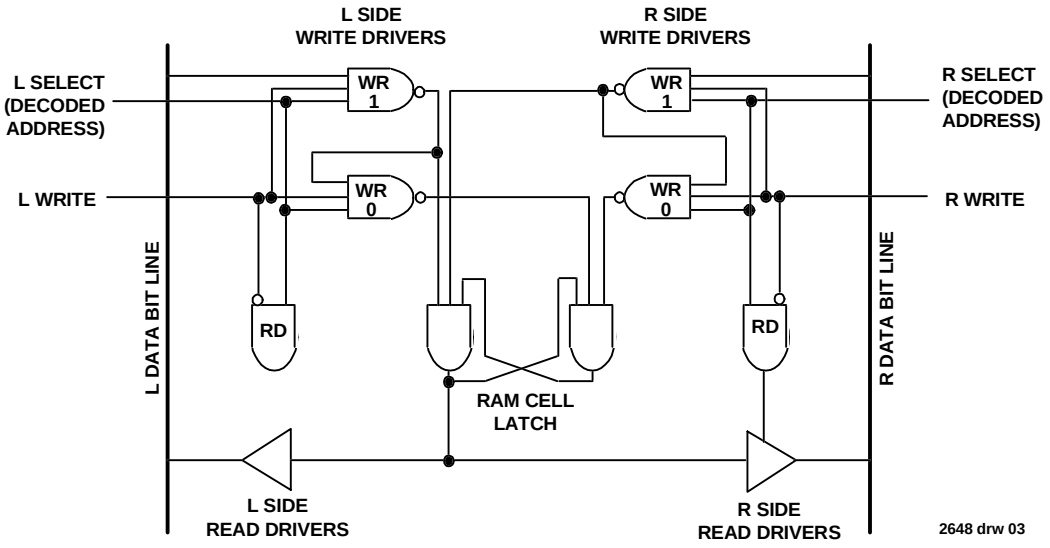


Figure 3. Dual-Port RAM Cell

directions. A common form of signaling is for one processor to cause an interrupt on the other processor. This allows the receiving processor to be informed of a communication without having to constantly check for it.

Hardware support for this signaling function is provided by interrupt logic, available on certain IDT dual-port SRAMs. A block diagram of this logic is shown in Figure 4. In these devices, the top two addresses of the memory chip also serve as interrupt generators for each of the ports. If the left side CPU writes into the even address of this pair (3FF in a 1K RAM) an interrupt latch is set and the interrupt line to the right hand port is activated. This interrupt latch is cleared when the right hand CPU reads from the even address. A similar set of logic is provided to allow the right hand CPU to interrupt the left hand one. This logic is associated with the odd address of the pair (3FE in a 1K memory). Providing this logic on chip saves the system designer from having to design in extra logic to allow one CPU to interrupt the other.

is because the probability of both processors using the same location at the same time is small. For example, if there are a thousand words in memory with a relatively uniform and random access of these locations by either side, the probability of a given location being accessed by one side is of the order of one part in a thousand. The probability of both sides accessing the same location at the same time is, therefore, of the order of one part in a million. As a result, the average throughput of the system is reduced by only one part per million due to dual-port SRAM access contention (again, assuming uniform random address access by both sides).

Busy Logic Design

Busy logic is called hardware address arbitration logic because it consists of hardware that decides which side will receive a $\overline{BUSY}$ signal

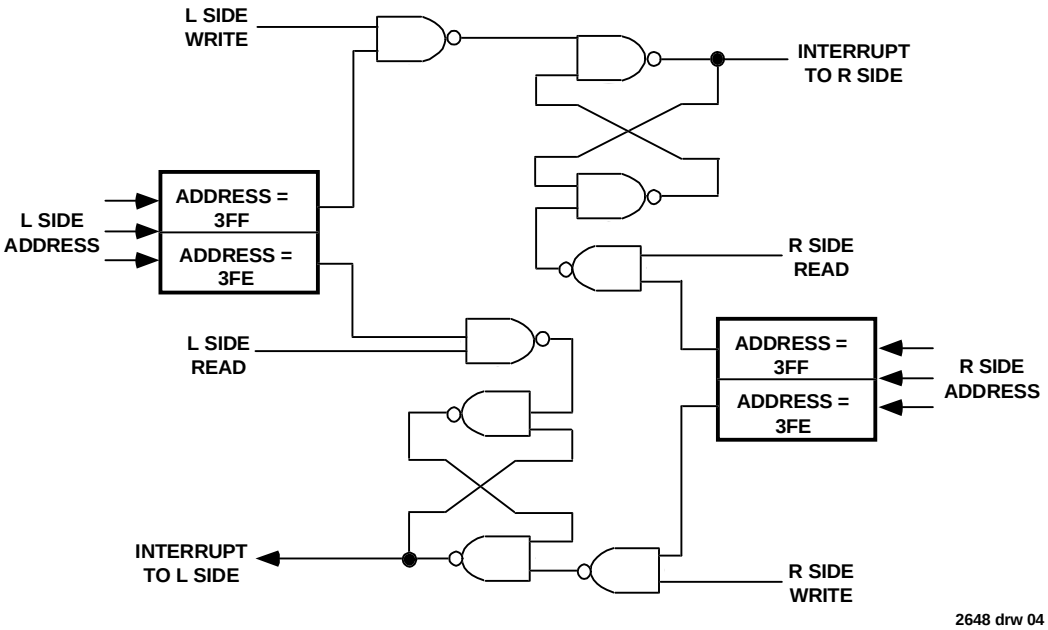


Figure 4. IDT7130 Interrupt Logic

Busy Logic Solves Interaction Problems

A problem can occur with dual-port memories when both ports attempt to access the same address at the same time. There are two significant cases: when one port is trying to read the same data that the other port is writing and when both ports attempt to write to the same word at the same time. If one port is reading while the other port is writing, the data on the read side will be changing during the read and corrupted data may result. If both ports attempt to write at the same time, the memory cell is being driven by both sides and the result can be a random combination of both data words rather than the data word from one side or the other. Busy logic solves this problem by detecting when both sides are using the same location at the same time and causing write inhibit only.

Note that although one or the other processor may have to wait occasionally, the throughput loss is minimal, typically less than 0.1%. This

if the addresses are equal. It consists of common address detection logic and a cross-coupled arbitration latch. A logic diagram of the type of $\overline{BUSY}$ logic used in the IDT dual-port RAMs is shown in Figure 5. The purpose of this logic is to provide a $\overline{BUSY}$ signal for the address that arrived last, to inhibit writing to the $\overline{BUSY}$ port and to make a decision in favor of one side or the other when both addresses arrive at the same time. This logic consists of a pair of address comparators, a pair of delay buffers, a cross-coupled latch and a set of $\overline{BUSY}$ output drivers. The address comparator output goes true when the addresses at its inputs are equal.

In the logic shown in Figure 5, the ability to detect which address arrived last is provided by the time delay buffers between address lines and the comparators. If we assume that the L address is stable and the R address changes to match the L address, the R address comparator will go true immediately while the L address comparator will become active some time later as determined by the time delay gates.

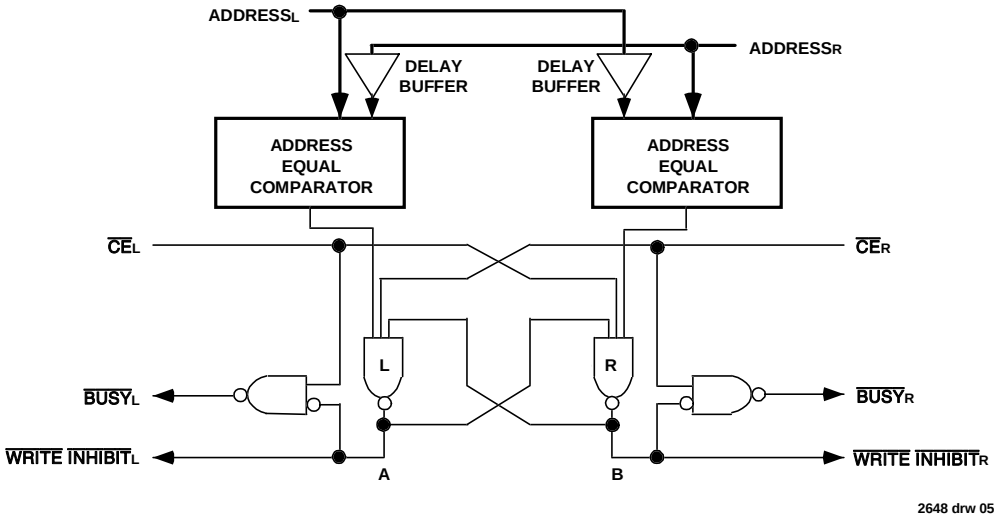


Figure 5. Dual-Port Busy Logic Design

The arbitration latch formed by the L and R gates reflects the address comparator output timing. This latch has three stable states, both latch outputs A and B HIGH, A LOW/B HIGH and A HIGH/B LOW. Initially, both A and B are HIGH because the outputs of both address comparators are LOW. We start with the L address stable and the R address arriving later. When the R comparator becomes active its output will go HIGH and B will go LOW. The A output will remain HIGH because its address comparator input will go HIGH sometime later and the L gate input from B output will go LOW before this occurs. The result is that the R gate B output will be active inhibiting writing to the R side of the dual-port RAM and activating the BUSY signal to the R port.

The extreme case of $\overline{BUSY}$ logic decision making is when both addresses arrive at exactly the same time. In this case, the outputs of both address comparators go HIGH at the same time activating both sides of the arbitration latch. The latch will settle into one of two states with either the A or the B latch output being active. The latch design ensures that a decision will be made in favor of one side or the other.

The chip enable lines come directly into the arbitration latch, although they could have been brought into the address comparators along with the other address lines. This is because if the chip enable for one side is inactive, both reading and writing for that side is automatically inhibited and/or arbitration is not needed. If the addresses are equal, the chip enable that arrives last will lose the arbitration. If both chip enables are active then arbitration will be determined by the settling of the address lines.

Temporary Assignment of Memory to One Side

A common problem in dual-port RAM application is the need to temporarily assign a block of memory to one side. For example, sometimes you need to update a data table as a whole and you cannot allow the other processor to use the table until you are done. This is called block allocation of the memory.

Block allocation can also be used to avoid the address arbitration problem since it is a way of ensuring that both sides do not use the same address at the same time. This method is also called software arbitration because the software on both sides decides and agrees as to who has permission to use a given portion of the memory. Software allocation has the advantage of not requiring $\overline{BUSY}$ logic, which is useful in systems which cannot accommodate a $\overline{BUSY}$ signal.

The design problem with block allocation is communication of the assignments between the CPUs. A simple but time consuming method is to pass messages between the CPUs, perhaps aided by interrupt logic. In the message method, processor A requests use of a block from processor B. Processor B agrees and sends permission back to processor A. When A is finished it sends a release message to B which responds with a release acknowledge to A. In this system, four messages are sent for each block assigned and released.

Semaphore Logic Support for Memory Assignment

Although block allocation is a software technique, it can benefit from hardware support. In message passing allocation, four messages must be passed to assign and release a block of memory. Semaphore logic, available in certain IDT dual-port RAMs, can be used to eliminate this message passing and its associated overhead. Semaphore logic provides a set of flags especially designed for the block assignment function. Each flag is used as a token to indicate which CPU has permission to use a block of memory.

Each semaphore flag can be set to one side or the other but not both. This ensures that only one side has permission to use the block of memory.

The IDT semaphore logic bits are designed to be used in a set-and-test sequence. Each bit is normally in the logic one state, indicating that it is not assigned to either side. A processor, desiring to assign a bit and, therefore, its associated block of memory, attempts to write a zero into the bit. It then reads the bit to see if it was successful. If it was, the bit will read zero, and the processor has use of the block. If it reads a one, it was unsuccessful, and the block is in use by the other side. The processor must then wait until the bit becomes zero, indicating that the other side has released it.

Semaphore flags have a particular requirement: a given flag can be

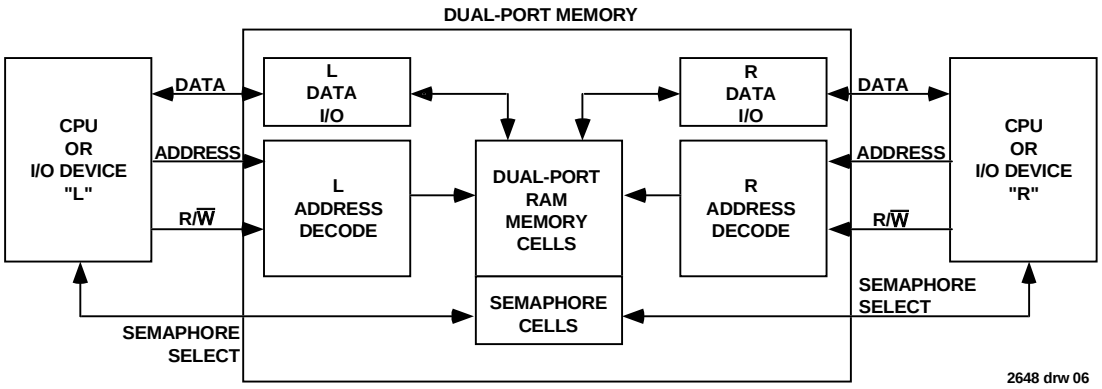


Figure 6. Dual-Port SRAM Semaphore Logic

assigned to only one side at a time. Specifically, you must not have a situation where both sides simultaneously think they have permission to use a block. Semaphore logic is designed to resolve this problem. If both sides attempt to set a semaphore flag at exactly the same time, only one side sees it set.

Semaphore flags consist of eight individually addressable dual-port latches. Each latch can be read and written by either side. They are selected by a separate chip enable, addressed by the three last significant bits of the address lines and are read and written through the D_0 data bit. Except for sharing the address, data and read/write pins of the RAM, the semaphore latches are completely independent, as shown in Figure 6.

A logic diagram of a semaphore logic flag is shown in Figure 7. In this logic, both flip-flops are initially at logic one and both grant outputs are HIGH. If only one flip-flop is set to zero, its corresponding grant output will go to zero. If the other flip-flop is set later, this will have no effect. If both flip-flops are set at the same time however, the latch will settle so that only one grant output goes LOW, ensuring that only one side receives permission to use the resource.

Dual-Port SRAM Chip Timing

The dual-port SRAM has a simple static RAM interface and timing

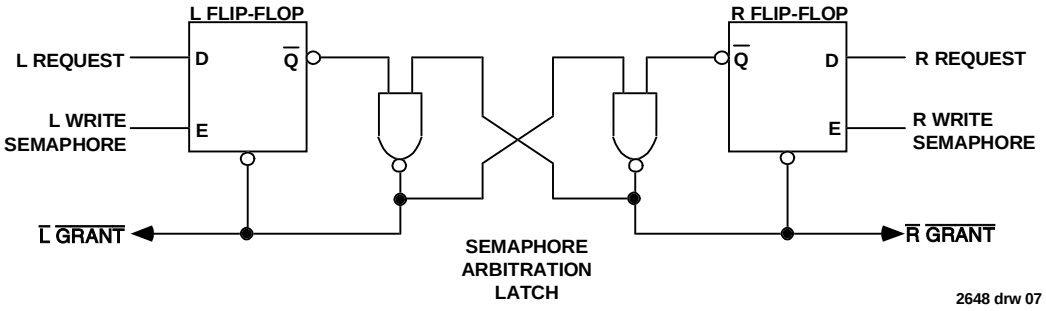


Figure 7. Semaphore Logic Design

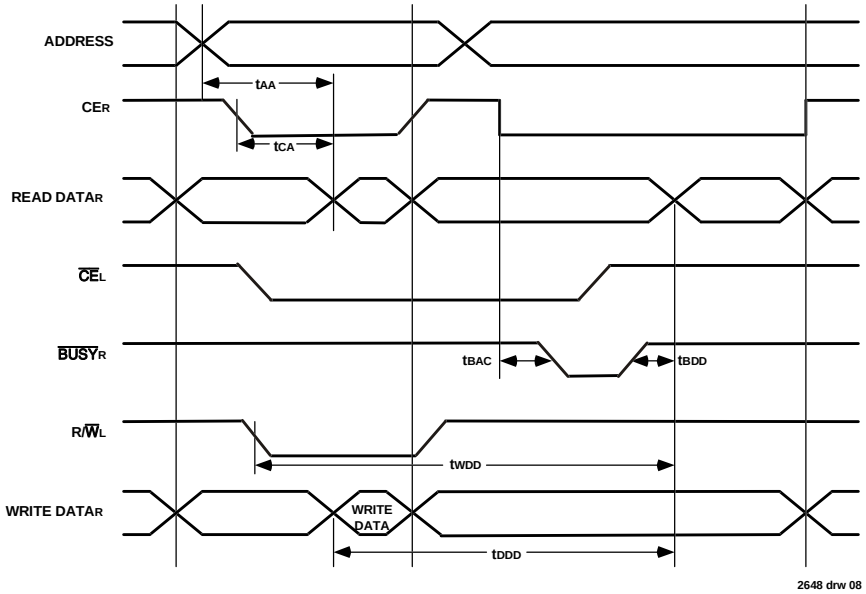


Figure 8. Dual-Port RAM Timing Diagram

requirements. There are some special requirements associated with BUSY, however. A timing diagram, shown in Figure 8, shows the relationships between address, data, read/write, chip select and BUSY signals for a dual-port RAM and BUSY logic. In this diagram, the chip select is used to enable the chip for a read or write operation after the addresses have settled. An arbitration is performed at the leading edge of the chip select.

Dual-Port Memory Expansion: Making Big Ones Out of Little Ones

Dual-port RAMs can be combined to form large dual-port memories. Expansion in memory depth with dual-port SRAMs is similar to expansion in depth for conventional SRAMs. An example of this kind of expansion is shown in Figure 9 where an 8K x 8 dual-port SRAM has been made

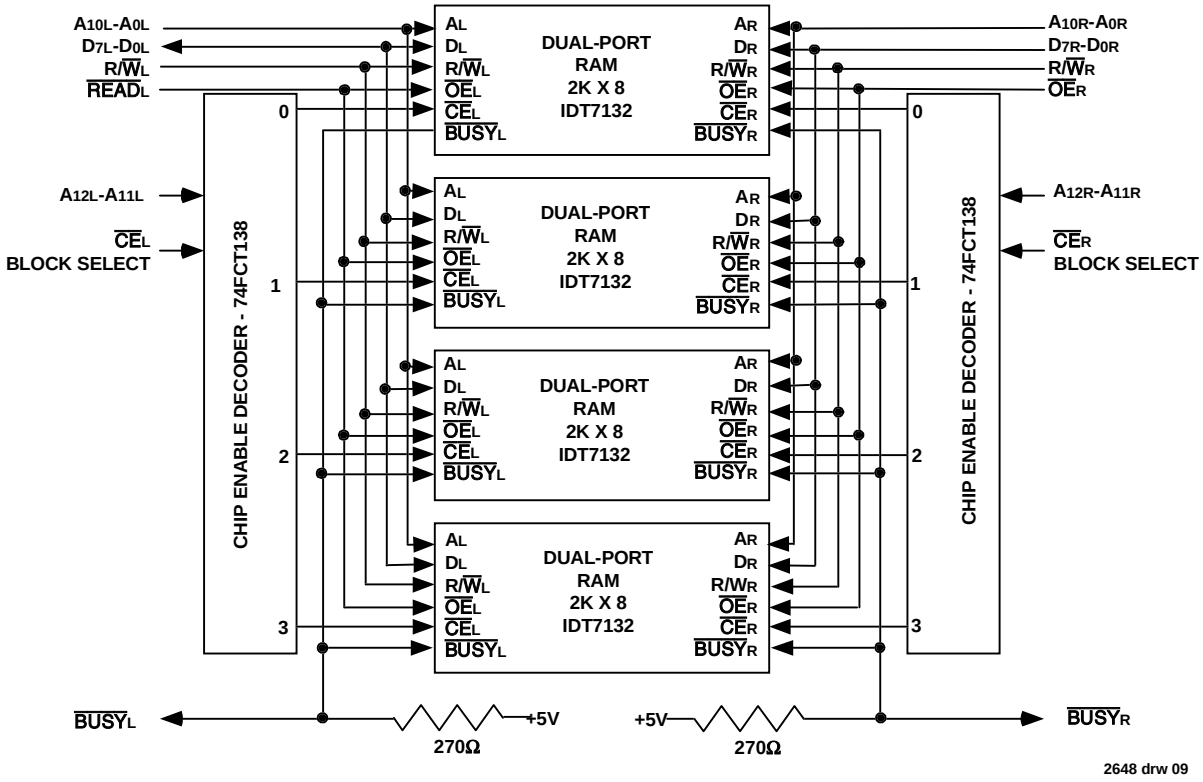


Figure 9. Depth Expansion of Dual-Port RAMs

out of 2K x 8 dual-port SRAM chips. It is important to note that the earlier dual-port SRAM designs have open-drain outputs for $\overline{BUSY}$, therefore pull-up resistors are required. The following devices require pull-up resistors: IDT7130/40, IDT7132/42, IDT33/43 and the IDT71321/421.

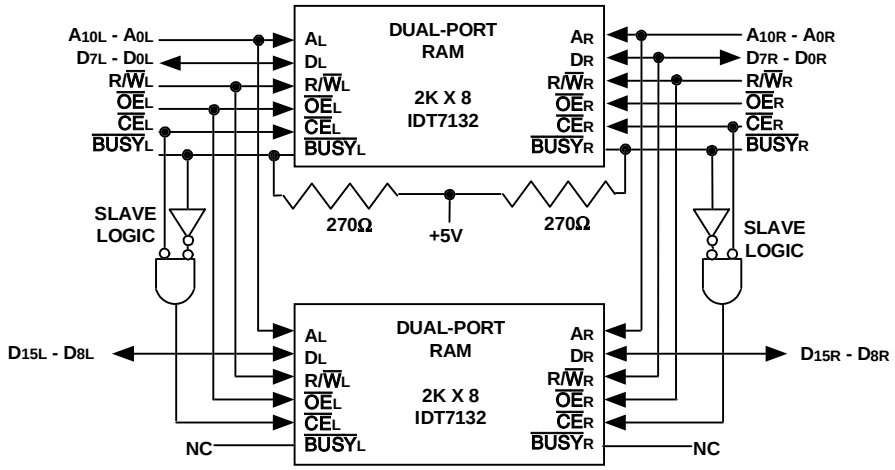
Width Expansion: The Busy Lock-up Problem

Dual-port RAMs can also be expanded in width. However, in this case, we have a subtle problem. Expansion in width implies that several dual-port RAMs will be active at the same time. This is a problem if several hardware arbitrators are active at the same time. If we examine the case of a 16-bit RAM made out of two 8-bit RAMs, we can better understand the problem. If the addresses for both ports arrive simultaneously at both RAMs, it is possible for one RAM arbitrator to activate its L $\overline{BUSY}$ signal and the other RAM to activate its R $\overline{BUSY}$ signal. If both $\overline{BUSY}$ signals are used on each side, we now have a situation where both sides are simultaneously $\overline{BUSY}$. The system is now locked up since both sides will be $\overline{BUSY}$ and both CPUs will wait indefinitely for their port to become free.

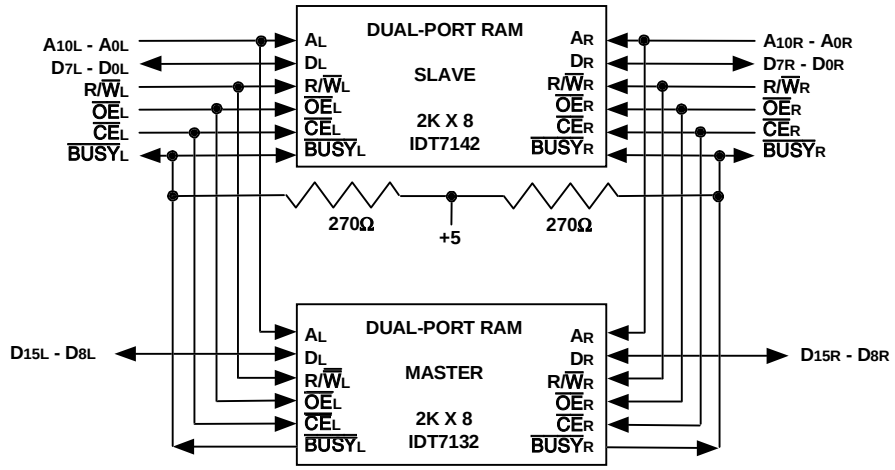
The $\overline{BUSY}$ Lock-up Solution: Use Only One Arbitrator

The solution to this $\overline{BUSY}$ lock-up problem is to use the arbitration logic in only one RAM and to force the other RAM to follow it. In this case, one RAM is dedicated as the arbitration master and additional RAMs are designated as slaves. Two solutions to this problem are shown in Figure 10. One solution is to add external logic to the chip-enables of additional dual-port RAM chips. The logic gates shown cause the slave RAM chip select to be disabled if the master RAM is $\overline{BUSY}$. Since only one set of arbitration logic is controlling the system the problem of slave lock-up is avoided.

The second, more desirable solution, is to use specially designed dual-port RAM slaves which are part of IDT's product line. These slaves incorporate the slave disable logic internally so that no additional logic is required to make a master/slave combination. In the slave, the $\overline{BUSY}$ pin serves as an input rather than an output. If the master activates $\overline{BUSY}$, the slave will sense this busy state and internally disable its write enable. Slaves provide a speed advantage over systems which use external logic



Width Expansion with MASTER Logic (Not Recommended)



Width Expansion with SLAVE Chips (Recommended)

2648 drw 10

Figure 10. Width Expansion of Dual-Port RAMs

to implement the slave function. Since the slave logic is built into the slave RAM, it can be designed so that there is no speed penalty when using slaves to expand the dual-port SRAM width.

Busy Logic Timing

In the case of address contention, the $\overline{BUSY}$ signal from the losing RAM port stabilizes some time after the leading edge of its chip select (or after its address settles, whichever comes last). If the $\overline{BUSY}$ signal is going to become active, it will become active during this time or not at all. If the $\overline{BUSY}$ signal is generated, the CPU must wait for $\overline{BUSY}$ to go away before completing the read or write cycle. Once the $\overline{BUSY}$ signal has gone HIGH the memory read or write cycle can proceed to completion.

Note that during the arbitration time following the chip select the $\overline{BUSY}$ signal may be changing. Since it is possible to have a glitch on the $\overline{BUSY}$ line during this indeterminate period, the $\overline{BUSY}$ line should be sensed as a level rather than as an edge.

$\overline{BUSY}$ arbitration will be somewhat slower in the extreme case where both addresses arrive at exactly the same time. This is because both gates of the arbitrator latch are initially inactive and must settle into a state where only one of them is active. There will be a period of time when both gates are in transition. This is called the metastable condition and is a classic and unavoidable problem in latch and flip-flop design. As a result, the $\overline{BUSY}$ settling time is somewhat longer in the low probability worst case than in the commonly observed typical case. The maximum arbitration times, t_{BAA} and t_{BAC}, on the data sheet give the worst case values, including metastability setting, for these times.

Read/Write Timing with $\overline{BUSY}$

The read and write timing for either port of the dual-port SRAM is the same as a simple static RAM in the absence of address contention. All the standard timing measures apply: read data address access time is t_{AA}, etc.

Dual-port SRAMs have additional timing specifications for the case of address contention where one port is busy and waiting for access. For the most general and conservative case, the read or write cycle for the waiting side should begin after the busy signal goes away. The actual timing can be somewhat shorter than this in most cases.

For the case where the waiting side is waiting to write, the write timing requirement is that the write pulse width be measured from $\overline{BUSY}$ going away. For the case where both sides are reading, the data will be available at the outputs one access time after the address/chip select lines settle even though the busy line is active. In the most common case, the trailing edge of $\overline{BUSY}$ will occur more than one access time after the address and data for the $\overline{BUSY}$ side have settled. As a result, the read access time as measured from the trailing edge of $\overline{BUSY}$, for this case t_{BDD}, is effectively zero.

The write/read case, waiting to read while the other side is writing to the same location, has some additional timing specifications. Since writing to a location by the L side, for example, will involve changing the data in the cell being read by the R side, there is a write-to-read propagation delay time. This time t_{WDD} for the delay for constant write data from the leading edge of the write pulse to the read data, and t_{DDD} for the delay for changing write data from a change of the write data to the read data.

If the writing side is running at minimum values for the write pulse or write data set-up times, the read access time, t_{BDD}, will no longer be zero. The actual t_{BDD} will be equal to t_{WDD} minus the actual write pulse width or t_{DDD} minus the actual write data set-up time, which ever is larger (and greater than zero). Note: t_{BDD} is always less than t_{AA} for the worst case of minimum write values. This is why the read or write cycle is begun from the trailing edge of $\overline{BUSY}$ for the most conservative case recommended above.

Width Expansion: Write Timing

When expanding dual-port SRAMs in width, the writing of the slave RAMs must be delayed until after the busy input at the slave has settled. Otherwise, the slave may begin writing while the $\overline{BUSY}$ signal is settling.

This is true for systems using slaves and for systems using conventional dual-port SRAMs with slave logic. This delay can be accomplished by delaying the write enable to the slave by the arbitration time of the master. This is shown in Figure 1 1.

Note that the write delay is required only in width expanded systems which use slave RAMs, not in single chip or depth expanded systems where only one chip is active at a time. This is because the individual devices have a built-in delay between the chip select and write enable inputs and the internal write enable to the RAM. Separate timing must be supplied in the slave case because this internal delay time can be balanced to the arbitration time only within a chip and can vary from chip to chip. If the delay time for the slave is less than the arbitration time of the master, writing could begin before BUSY became active, as above.

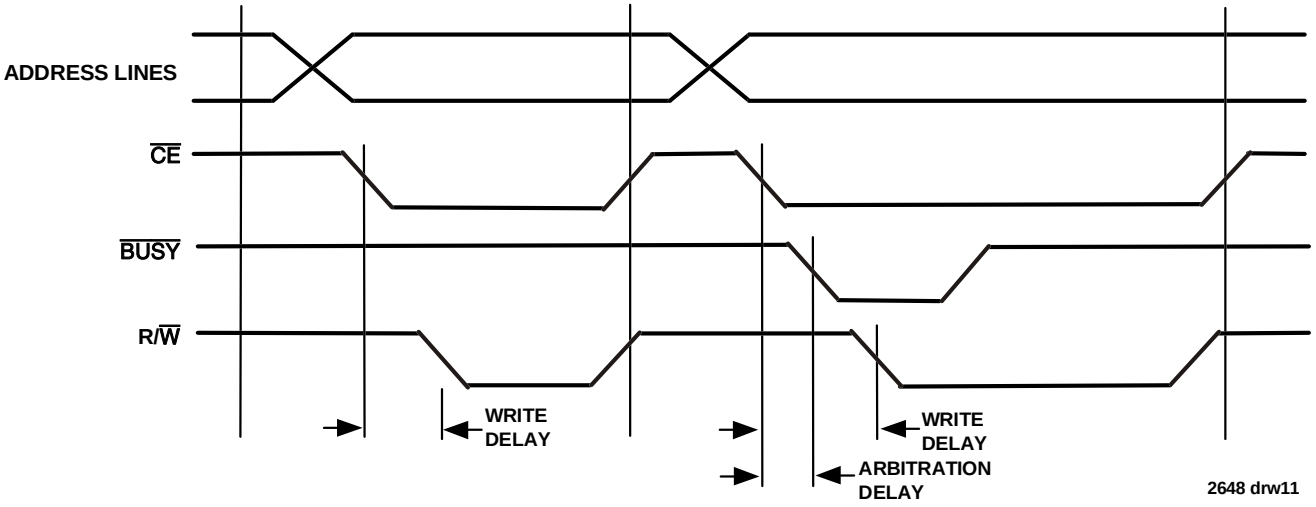


Figure 11. MASTER/SLAVE Write Timing

Width and Depth Expansion: An Example

These techniques for expanding dual-port memories in width and depth are combined in the example shown in Figure 12. In this example, an 8K x 16 dual-port memory is made from 2K x 8 dual-port SRAMs in master/slave combination.

Using Them: Dual-Port RAM Application Examples

Examples of dual-port RAMs used for CPU-to-CPU communication are shown in Figures 13, 14 and 15. In Figure 13, a pair of 8-bit processors communicate using a single 2K x 8 dual-port SRAM chip. In Figure 14,

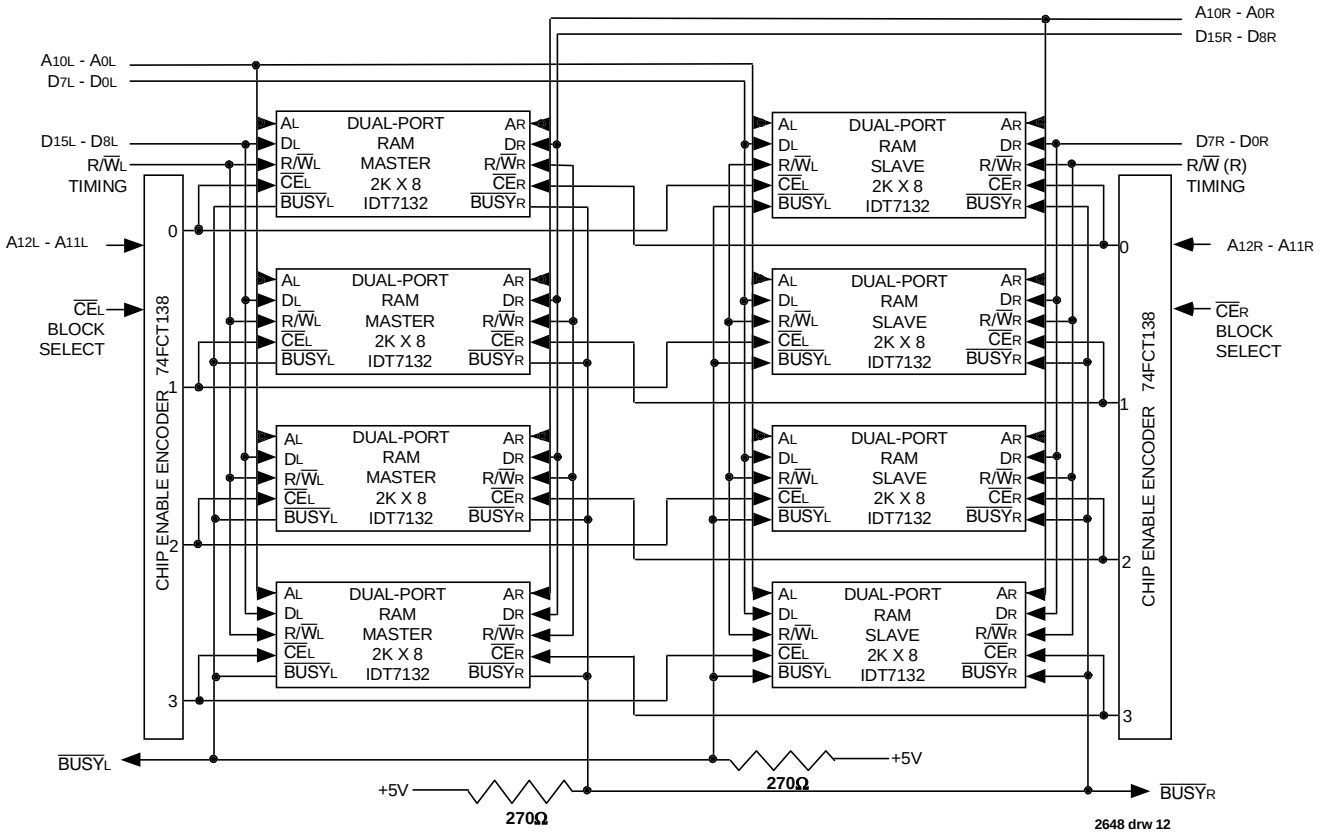


Figure 12. Width and Depth Expansion of Dual-Port RAMs

there is a similar system where a pair of 16-bit processors communicate using a pair of dual-port SRAMs and a master/slave configuration. Finally, in Figure 13, we have an 8-bit processor communicating with a 16-bit processor through two 2K x 8 dual-port SRAMs.

In Figure 13, two Z80 microprocessors communicate using a single IDT7132 dual-port SRAM. The IDT7132 is controlled by the chip enable. The write enable is set up in advance by the WR signal from the Z80 and the chip enable is used to write data into the RAM or to gate the read data onto the Z80 bus. The output enable (not shown) is tied to ground (continuous enable). The write enable is used to disable the output drivers.

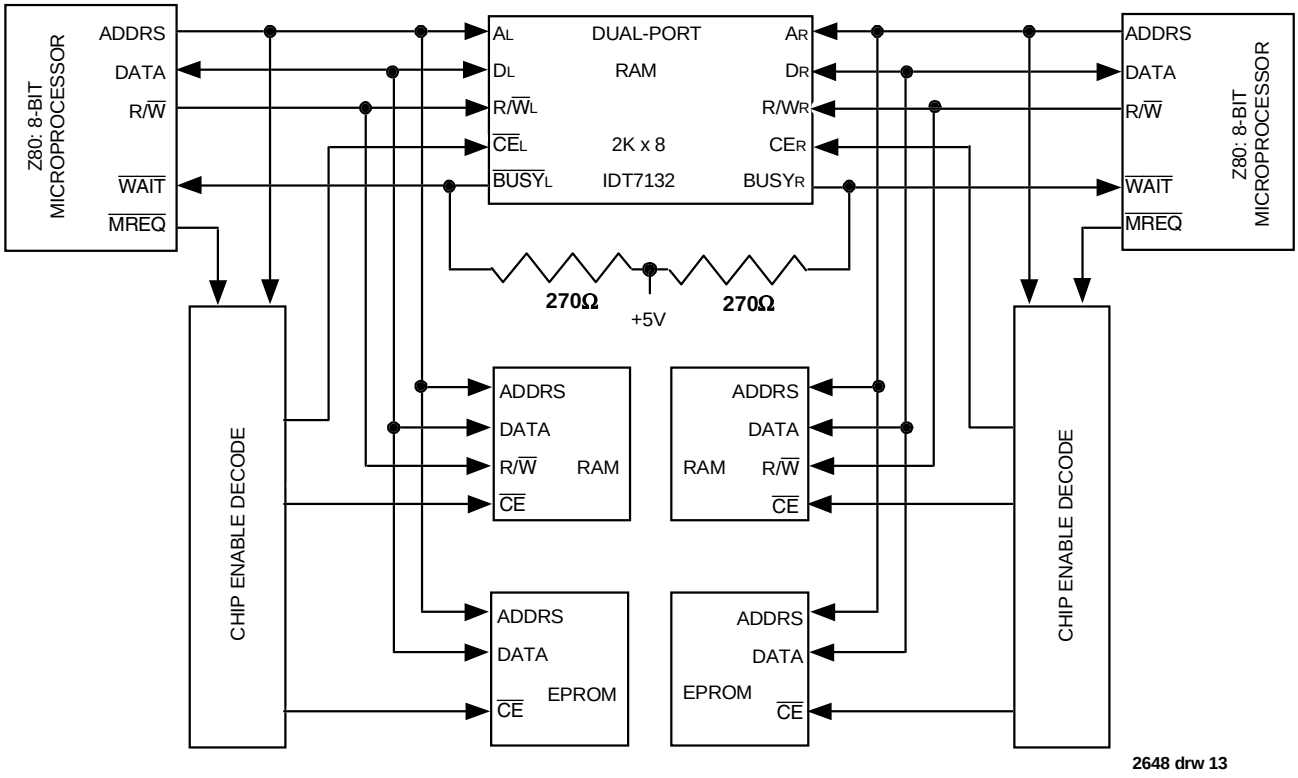


Figure 13. 8-bit to 8-bit CPU Communication

In Figure 14, two 68000 microprocessors communicate through a pair of dual-port SRAMs. A IDT7132/7142 master/slave pair is used to avoid the busy lock-up problem. Note that the Address Strobe ($\overline{AS}$) from each 68000 is used with an address decoder to enable the dual-port SRAMs. This is to maintain the address for read-modify-write cycles so that arbitration is not lost between the read and the write. This is important for test and set instructions, for example.

In Figure 15, a Z80 and a 68000 communicate using a pair of IDT7132 dual-port RAMs. No slave logic is required because the Z80 side chip enable decode ensures that only one dual-port SRAM will be enabled at a time. Otherwise, this figure is a combination of the logic from Figures 13 and 14.

Summary and Conclusion

The development of true dual-port memories in integrated circuit form provides the designer with the ability to set up communication between components of a computer system while avoiding many of the problems of prior systems. While the concept of dual-port memory has been with us from the early days of computing in the form of DMA, the new dual-port ICs can provide this function at very high speeds and without the delays associated with earlier designs. Because of the utility of the dual-port memory concept these chips should come into wide-spread use and become one of the standard components used by the computer designer.

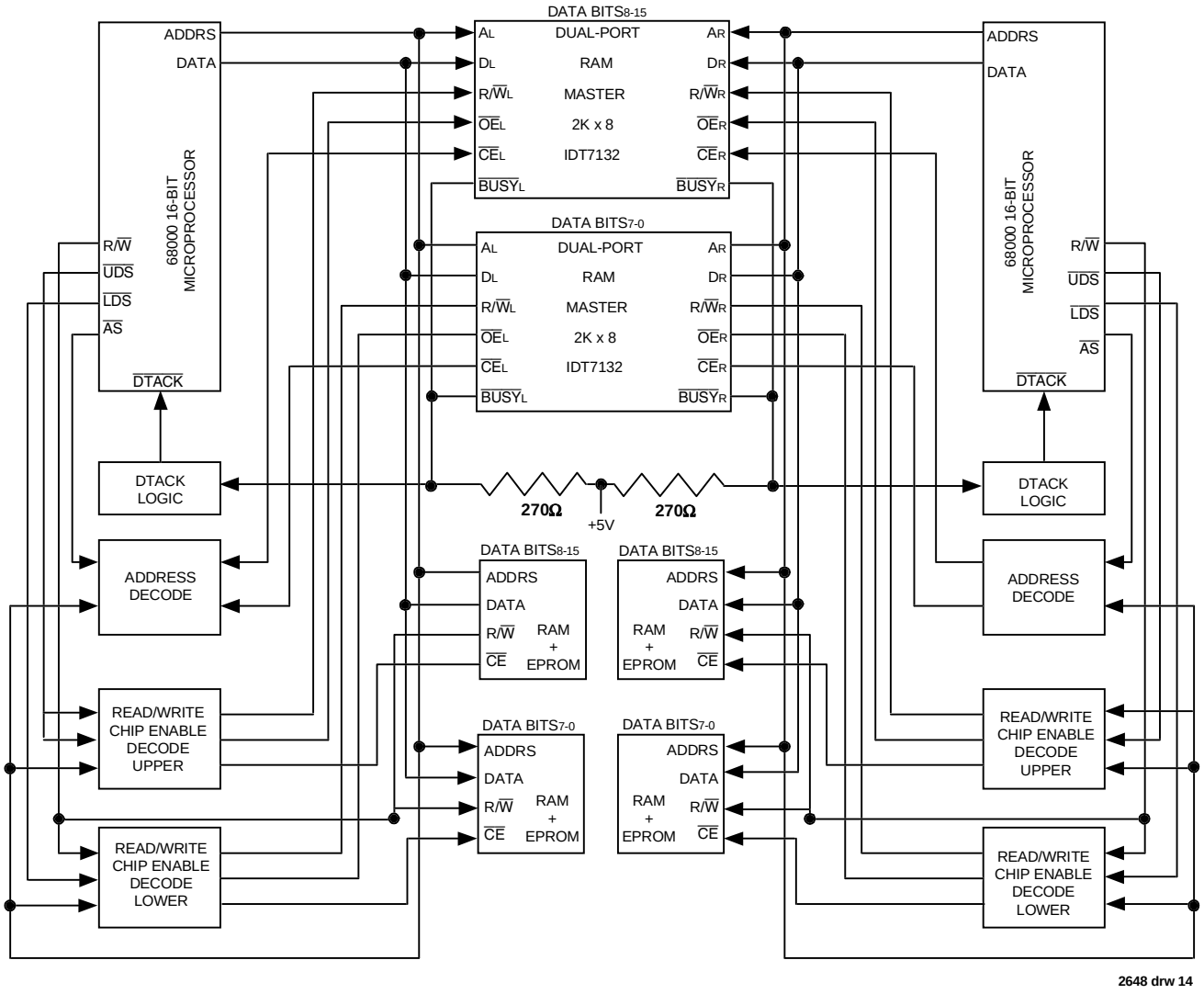


Figure 14. 16-bit to 16-bit CPU Communication

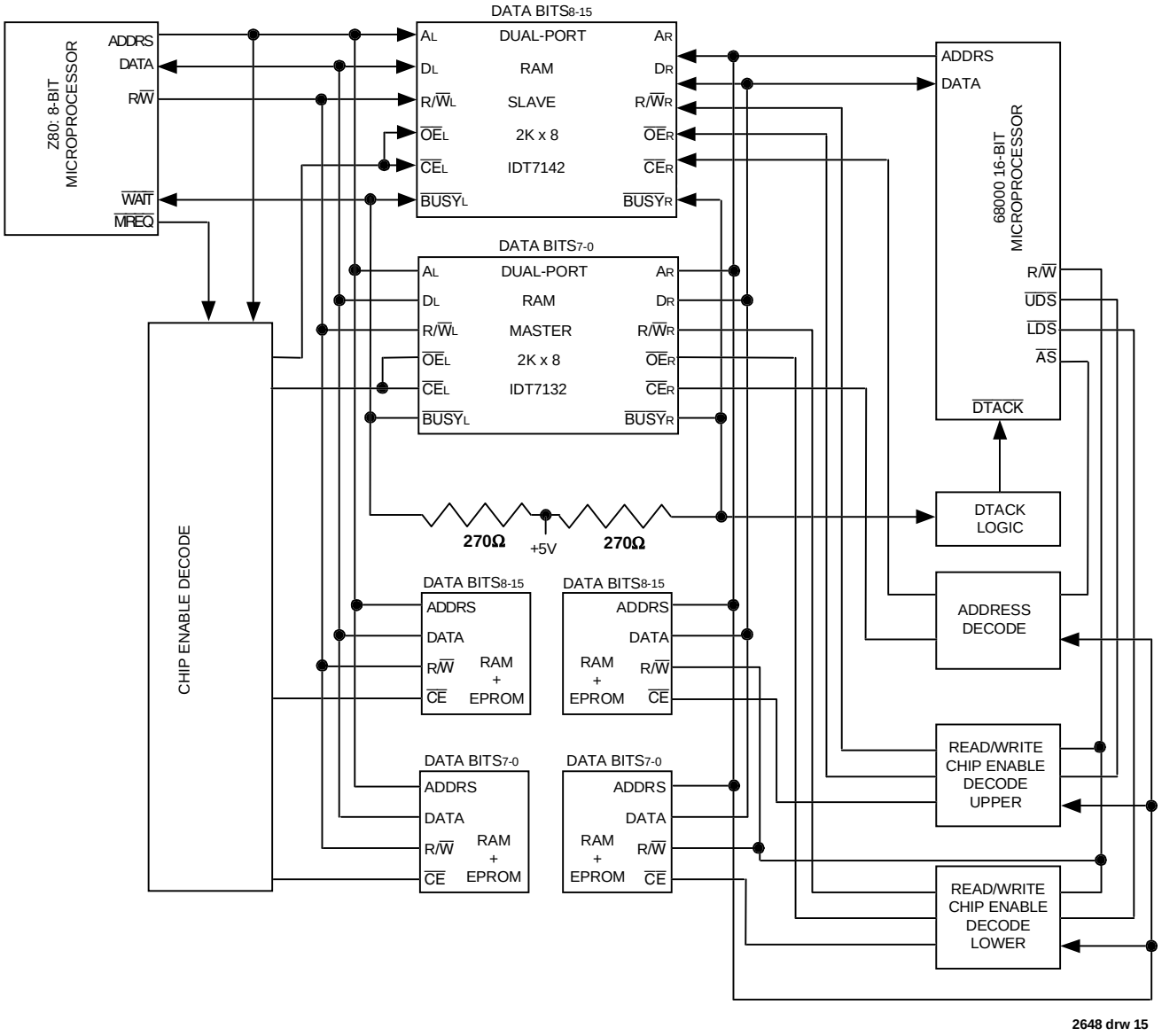


Figure 15. 8-bit to 16-bit CPU Communication

CORPORATE HEADQUARTERS
2975 Stender Way
Santa Clara, CA 95054

for SALES:
800-345-7015 or 408-727-6116
fax: 831-754-4608
www.idt.com

for Tech Support:
831-754-4613
DualPortHelp@idt.com