THE MOST COMMONLY ASKED QUESTIONS ABOUT ASYNCHRONOUS DUAL-PORT SRAMS

APPLICATION NOTE AN-91

By Mark Baumann and Cheryl Brennan

What is a dual-port SRAM?

A dual-port SRAM is exactly what it sounds like. It is a single static SRAM array accessed by two sets of address, data, and control signals.

What is an asynchronous dual-port?

An asynchronous Dual-Port is a dual-port that responds to address and control pin inputs without the need for clocks or counters. These devices allow simultaneous access to a single static SRAM memory location from two busses. Most asynchronous dual ports have arbitration logic.

How is simultaneous access conflict handled?

When designing with a dual port SRAM it becomes obvious that there are a few instances in which one side could have a potential conflict in accessing the data. These potential conflicts occur in an IDT multi-port SRAM only when two or more ports are trying to access the *exact same cell* in the SRAM array. If the locations are even one location different the conflict will *not* occur. And to take it one step further, it will only be a problem if one or more of the ports is trying to *write* to this common cell. At this point the integrity of the data at this location could be compromised.

There is no contention or confict when both ports are reading. Contention only occurs when one or both ports are writing.

The two instances which must be considered are (a) one port reading and one writing, and (b) both ports writing to the same location simultaneously. In both these cases, there is the potential that if indiscriminate access was granted to the location from both sides, the data being read in the case of a write/read could be the old data, new data, or any combination of the two. It is interesting to note that the write will complete and at the end of this cycle the cell will contain the correct written data, as long as all specified timing is met and a "slave" device is used. In the case of two writes the data which will be written into the cell is indeterminate since the data from the two sides will collide in the cell. No cell damage occurs and, if one port were to remain a valid access time longer than the other, this port's data would be contained in the cell.

To contend with this type of conflict three types of logic are available:

a) Busy Logic.
b) Interrupt Logic.
c) Semaphore logic.

The following paragraphs will explain how each of these functions works and the advantages and disadvantages of each.

Busy Logic (master device)

Busy Logic provides a write block to the losing port when the two ports of the dual port are accessing the exact same cell. What is happening internally is that a scheme which arbitrates between the two sides is

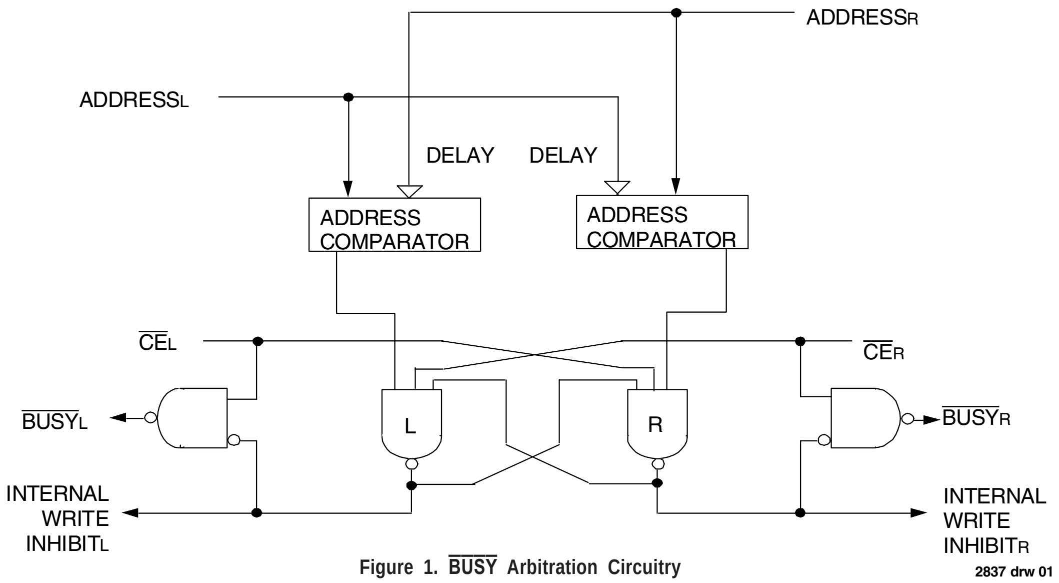

Figure 1. $\overline{\text{BUSY}}$ Arbitration Circuitry

MARCH 2000

implemented. This arbitration is dependent on *only address* and $\overline{CE}$ and *not* on R/$\overline{W}$. Figure 1 depicts the logical implementation of this scheme. Note that the side which loses the arbitration, and receives the $\overline{BUSY}$ signal will be internally prohibited from performing a write if it was attempting to do so. As stated earlier, an attempted read will never be prohibited even if a $\overline{BUSY}$ is received.

In the decision process there is a window of 5ns in which if one side has a valid address and the $\overline{CE}$ signal 5ns before the other side, then this earlier arriving side is guaranteed to "win", forcing the opposite side to receive a $\overline{BUSY}$ signal. However, if the two sides receive a valid address and $\overline{CE}$ within this 5ns window, it is *indeterminate* as to which side will receive a $\overline{BUSY}$ signal. One side is guaranteed to receive it within the specified t_{BAA} and/or t_{BAC} times.

This part is necessary in a situation where the two ports of the RAM are running asynchronously with the potential of accessing the same cell, and you cannot suffer the potential corruption of data during this collision. The other forms of logic (semaphore and interrupt) can handle situations where this shared memory can be allocated time slots so that collisions can be avoided by careful allocation of resources.

Interrupt Logic

The devices which have interrupt logic integrated behave in the following manner. The upper two address locations are designed to act as a form of mailbox. When the left side writes to the highest dual port location, a normal write occurs at this location. An interrupt signal ($\overline{INT}$) is subsequently set on the right side. The setting of this signal has absolutely no impact on the operation of the dual port itself, and will not prohibit any operation from occurring to the RAM itself. It is strictly used as a software tool to pass a signal from one side of the RAM to the other, or from one processor to another. To send an interrupt from the right side to the left, a write operation must occur to the next to uppermost location. In either case the interrupt is cleared by a read of the opposite port's interrupt location. It is important to note that the read will clear the interrupt signal but the data in this location is undisturbed. Figure 2 depicts the logical implementation of this function. As seen in this figure, the actual interrupt signal is arbitrated so that if a read and write occur at the same time a BUSY arbitration scheme is invoked. It is also important to note that on many of the devices with this function the output is an open drain and will require some value of pull-up resistor to properly drive the line. This applies to the IDT7130/7140, IDT7133/7143, IDT7132/7142 and the IDT71321/71421.

A second point of interest is that a write to the opposite port's interrupt location will not set the interrupt flag nor will it clear the interrupt flag if it had already been set by the other port.

Another point of interest is that if the Interrupt locations are not used the two upper bytes are treated as normal memory address locations.

Semaphore Logic

The semaphore logic which is built into some of IDT's dual port RAMs is general purpose logic. This means that the logic has no direct connection to the functioning of the RAM-much like the interrupt logic (see Fig.3). The features of the semaphore are as follows:

a) They consist of eight individual locations in the device.

b) These are not RAM locations so they do not take away from any usable space of the device.

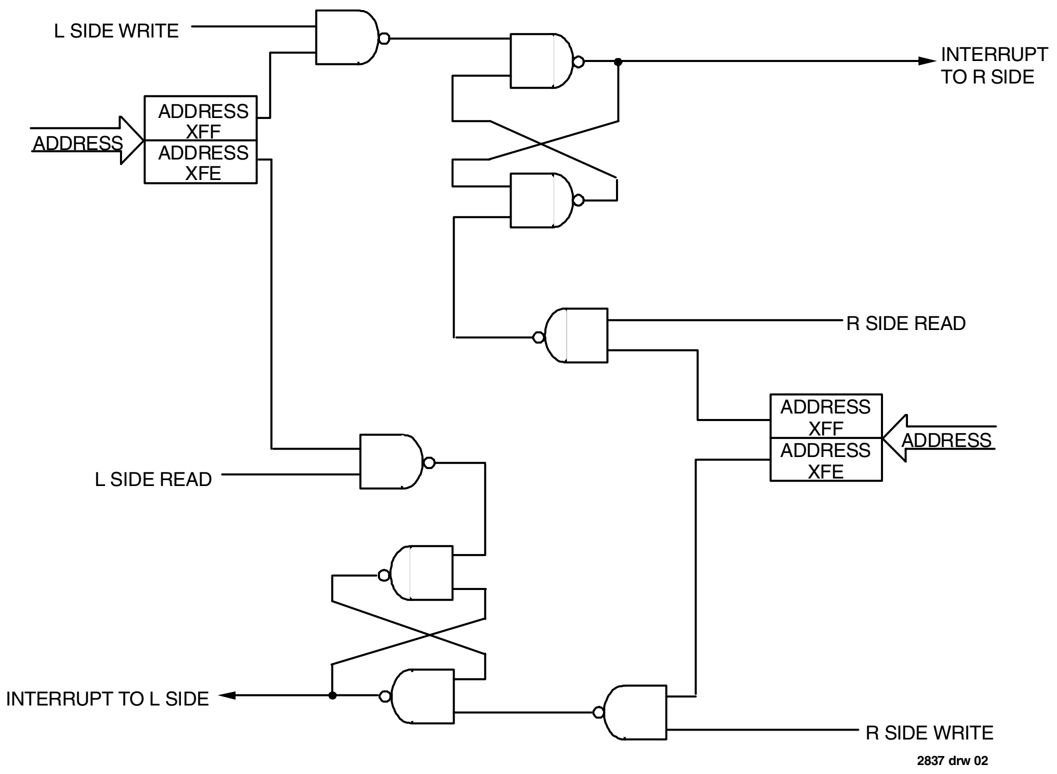

Figure 2. Interrupt Logic

c) Each of these locations has its own arbitration to prevent the possibility of the two ports of the RAM getting access to the semaphore or software flag simultaneously.

d) Each location is accessed over the standard data bus using the lower order address bits A_0-A_2 to address the eight locations and a $\overline{SEM}$ pin. $\overline{CE}$ must *not* be active to access these locations since they are not RAM locations.

e) To set a flag: write “1” on D_0 only (other bits are don’t cares), a write must occur to the semaphore location, and be immediately followed by a read of the same location, to insure that the write was not blocked by the opposite port.

f) To insure that the request was granted, the semaphore location must be read. To do this the $\overline{SEM}$ line and the corresponding A_0-A_2 lines must be driven and $R/\overline{W}$ must be HIGH. At this time the state of the flip-flop will be driven on all eight data lines of this port. If the write of a “1” was successful when a read is performed the data out will be FF(hex). If the write was unsuccessful or if you are trying to reset the location with a write of a “0” when the read is performed the data presented on the outputs will be 00(hex).

g) Once a port gains possession of this flag or semaphore location the only way for the opposite port to gain access to this flag is for the port with possession to clear this location by writing a “0” to this location. At this time this location can be arbitrated for by either port.

Because the implementation of this hardware does not directly affect the RAM array itself, it is possible for the designer to define in software what each flag represents. This could mean something as simple as dividing the RAM into eight blocks and requiring a valid set and test operation (semaphore access) to occur before an access is granted to one of the sections. Or it could be used as a flag for any other function in the system design, freeing up space which may otherwise be taken up with additional registers and logic.

If I don't use the interrupt can I still use those upper addresses in the array?

As mentioned under interrupt logic,these two upper address locations are standard RAM locations. They can be used as such,the only side effect being that the interrupt flag will be set and cleared during accesses to this location. If these lines are not being monitored the user will not be subject to any side effects from this function being on the device.

What is the difference between a master and a slave device?

The slave device was designed to address a problem which can arise if there is a need to expand these devices in width. It is important that you understand the problem which can occur if two or more master devices (devices which have arbitration) are used in width expansion. There is the potential that the two devices will arbitrate differently and a BUSY signal will be sent to both sides of the system (one from each of the two devices). This could potentially lock the system. It is necessary to address this problem with a device that does not drive a BUSY signal but can accept one as an input from the one master device which performs the arbitration. This will have a direct effect on the speed at which a R/W can be issued to these devices. It is important to insure that this concern is addressed by the system designer, due to the catastrophic danger of this lock-up condition.

How should I use the $\overline{\textbf{BUSY}}$ line?

The $\overline{BUSY}$ line’s main purpose is to control simultaneous access conflicts. One design consideration of using the busy output is to make sure that you *do not* put an edge sensitive device on this output. The $\overline{BUSY}$ output will be noisy during an arbitration. It is therefore necessary that this output should be sampled after a valid t_{BAA} and/or t_{BAC} time. This is the time at which the dual port guarantees a stable $\overline{BUSY}$ output on the device. Prior to this it is possible that this output is in a state of flux.

How do you handle depth and width expansion?

The best way to approach this may be to take each case individually. This does not preclude the fact that these devices can be placed in an array in which both depth and width expansion is performed, exactly like a standard SRAM array.

Depth expansion

Figure 4 depicts the connections required to expand a device like this in depth. It is very straightforward, and is handled exactly as you would a standard static RAM, with the exception of the $\overline{BUSY}$ outputs, which must be tied together to a pull-up resistor and to the input of whatever circuitry is used to generate wait states on either side.

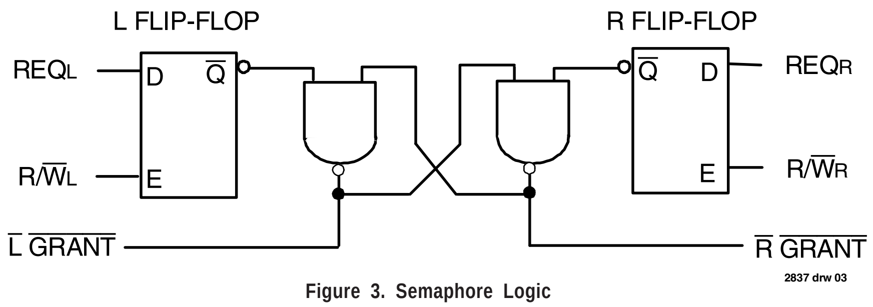

Figure 3. Semaphore Logic

Width expansion

Figure 5 depicts the connections required for width expansion. Here again, the devices are connected exactly as you would a standard static RAM with the one major exception being the use of one master device and one or multiple slave devices. This need arises because in this configuration, if two master devices were used, both ports could potentially receive a $\overline{\text{BUSY}}$.

If I use the Busy Logic, what is the actual timing I must use before issuing a R/$\overline{\text{W}}$?

In using a dual port device without width expansion, the access time is as specified in the data sheet. A valid write will occur in the RAM, it knows $\overline{\text{BUSY}}$ was received on this side, and the proper setup and hold times

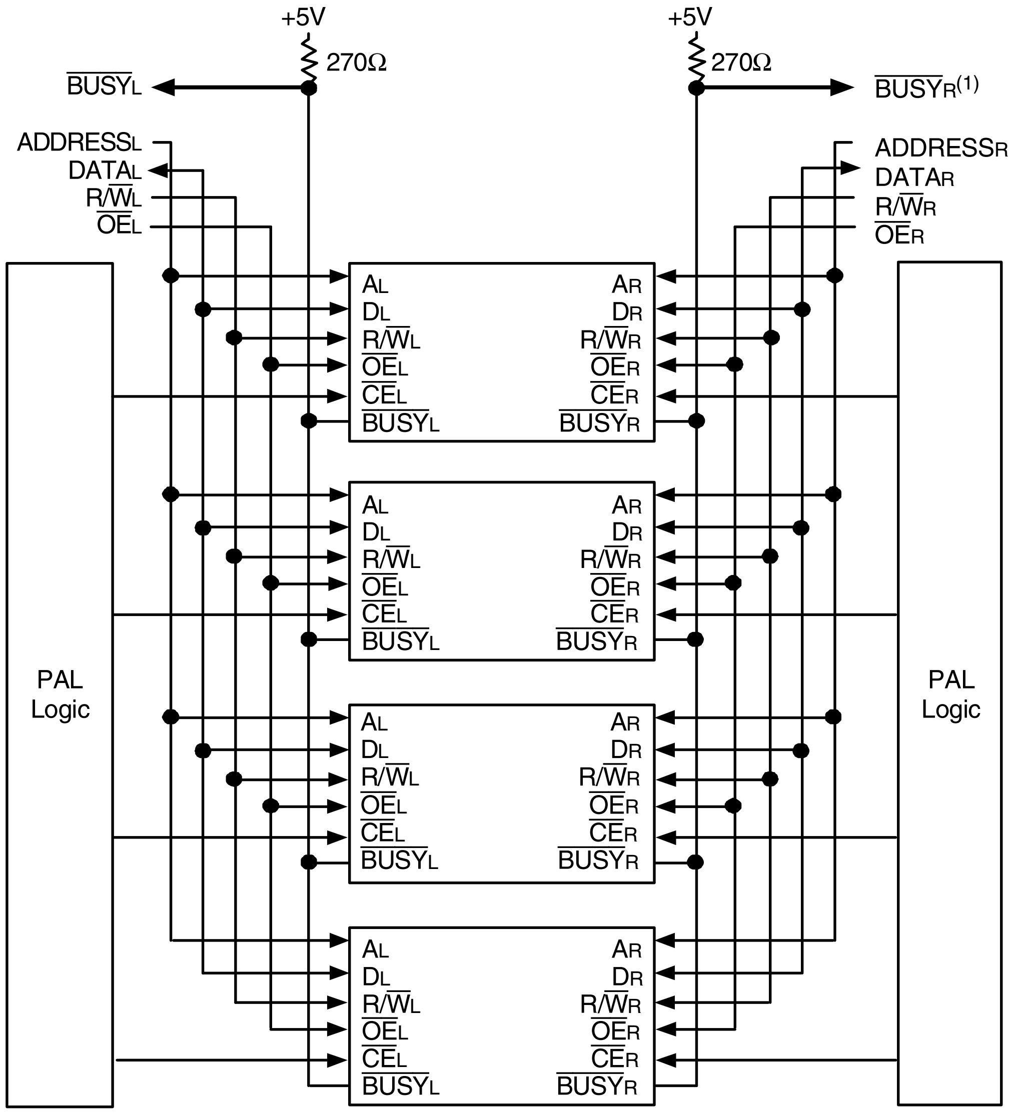

Figure 4. Depth Expansion

NOTE:
1. $\overline{\text{BUSY}}$ signals may be tied together with a pull-up resistor on devices equipped with open-drain outputs (IDT7130/7140, 7132/7142, 71321/71421, 7133/7143). On devices equipped with totem-pole outputs, $\overline{\text{BUSY}}$ signals must not be tied together, but instead monitored via a combination of AND gates.

as well as tWP and tDW times are met. If however, a $\overline{\text{BUSY}}$ is received, the write time access starts to occur at the time the $\overline{\text{BUSY}}$ signal goes away. It can be considered that a valid access starts as soon as $\overline{\text{BUSY}}$ goes back HIGH. Therefore, a read access takes tAA after $\overline{\text{BUSY}}$ goes HIGH. *A valid cycle time or tWC starts after $\overline{\text{BUSY}}$ goes HIGH.*

As far as initiating a write cycle there are no special considerations on a memory array not using width expansion. The R/$\overline{\text{W}}$ signal can go active any time during the tWC time as long as tWD and tDW are met. If a R/$\overline{\text{W}}$ signal is activated at the beginning of a cycle it will be internally held off for tBAA. This will prohibit any potential corruption of the addressed cell during $\overline{\text{BUSY}}$ arbitration.

If the RAMs are being used in a width-expanded mode, the timing of a write cycle must be modified to compensate for the timing of the master/slave interface of the dual-port RAM. In essence, what must happen is that the issuance of the R/$\overline{\text{W}}$ signal must be delayed to allow for the RAM to issue the $\overline{\text{BUSY}}$ signal. This is because the slave device, used in width expansion, does not have an internal hold-off mechanism as the master device does. It is totally dependent on the master device to drive the $\overline{\text{BUSY}}$ input, which in turn, disables the write to the RAM, for the port which receives the $\overline{\text{BUSY}}$. Figure 6 depicts the timing required for a width expanded array.

What parts are pin compatible for up grades in depth?

It is important when designing for this type of upgrade that all functions be investigated—especially with the older 1K, 2K, and 4K densities. These devices have the same package and almost the same pinout. The differences are in the functions associated with the device. The 1K PDIP has both $\overline{\text{BUSY}}$ logic and interrupt logic. The 2K device in the same package eliminates the interrupt pins and uses these pins for the extra address lines and the 4K devices eliminates the $\overline{\text{BUSY}}$ logic to add the next address line. Even more changes exist in the PLCC and TQFP versions of the device, even though the footprint of each of the DIPs is identical, the designer must be willing to trade off functionality for density in the same footprint.

Therefore, in answering this question only footprint and density were considered, special functions like Interrupt, Semaphore, and $\overline{\text{BUSY}}$ logic were not. Here are your options (5V and 3.3V devices have equal capabilities):

- 7130/40 (1Kx8) PDIP is able to be upgraded to the 7132/7142 (2Kx8) which is able to be upgraded to the 7134 (4K x 8).

- 7130/40 (1K x 8) PLCC is able to be upgraded to the 7132/7142, or 71321/71421 (2K x 8), all of which are able to be upgraded to the 7134 and 71342 (4K x 8).

- 7130/40 (1K x 8) TQFP is able to be upgraded to the 71321/71421 (2K x 8), all of which are able to be upgraded to the 71342 (4K x 8).

- 7005 (8K x 8) PLCC is able to be upgraded to the 7006 (16K x 8), 7007 (32Kx8), and the 7008 (64Kx8).

- 7005 (8K x 8) TQFP is able to be upgraded to the 7006 (16K x 8), 7007 (32Kx8), 7008 (64Kx8), and the 7009 (128Kx8).

- 7133/7143 (2K x 16) is a standalone device.

- 7024 (4K x 16) PLCC is able to be upgraded to the 7025 (8K x 16) and 7026 (16Kx16).

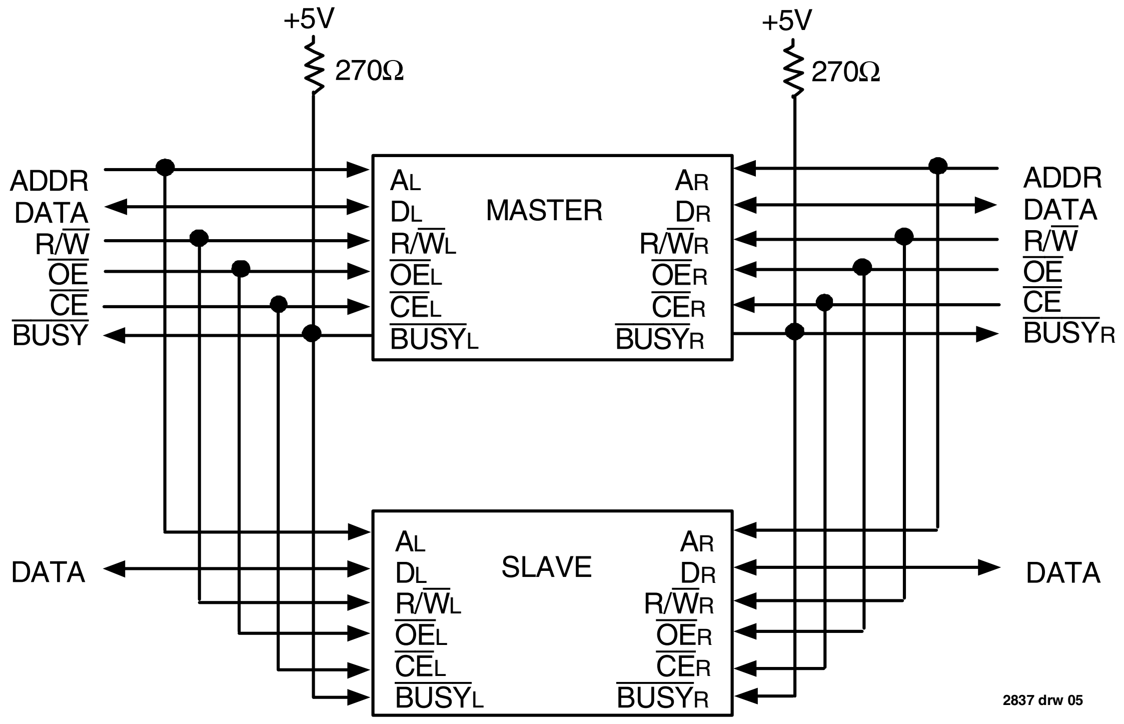

Figure 5. Width Expansion

• 7024 (4K x 16) TQFP is able to be upgraded to the 7025 (8K x 16), 70261 (16Kx16), 7027 (32Kx16), and the 7028 (64Kx16).

As with any summary, this is a snapshot in time and could be continually updated.

Are there any special layout considerations I should take in using a device in which 16 or 18 outputs can be switching at once?

In short, YES.

The first thing to remember about any of IDT's devices is that they are CMOS devices. Because of this, a number of standard design practices should be followed, some of these are as follows:

1) **Do not leave any inputs floating or undetermined.** This means that all inputs are recommended to be either driven at all times or pulled through a resistor to VCC or GND. This is even true if the device is driven by a tri-state driver that will tri-state for any period of time. If this practice is not followed, both the pull-up and pull-down stages on the input of the device can turn on simultaneously, causing the device to draw an extreme amount of current and/or causing the device to oscillate.

2) **Use a good decoupling scheme for the device.** Here again, if the specifications of the device are examined closely, it is clear that, when enabled, the almost instantaneous current requirement of the device can go from as little as a few hundred micro amps to close to 200mA. With this type of current requirement it is necessary to consider the value of the capacitor (standard values like 0.1µf and 0.01µf work well most of the time), as well as the frequency response of the capacitor. It is best to decouple at least every device and, if the device has multiple VCC inputs, one for each of these is ideal. We are sure that there are a few who would like to argue over where the capacitor should be placed, near the VCC or near the GND. We are merely suggesting that decoupling should be used as liberally as possible.

3) Somewhere on the board there should be some bulk decoupling such as a large tantalum capacitor. This is usually placed near the power input for the module and helps decouple low frequency backplane and power supply noise from the board.

4) VCC and GND planes are still considered a recommendation but should be considered a necessity. The planes are needed as a source and sink for current as well as providing a means of controlling etch impedance,and a means of isolating critical signals from cross-coupling.

5) Capacitive loading effects on address access time is a tricky question. In general, the RAM-based products from IDT tend to derate at an average of .05ns/pf over the specified data sheet load. In the case of the CMOS RAMs this is 30pF. So if an output on your RAM has 50pF of load the access time would be [t_{AA} + (50-30)*.05] = (t_{AA}+1ns).

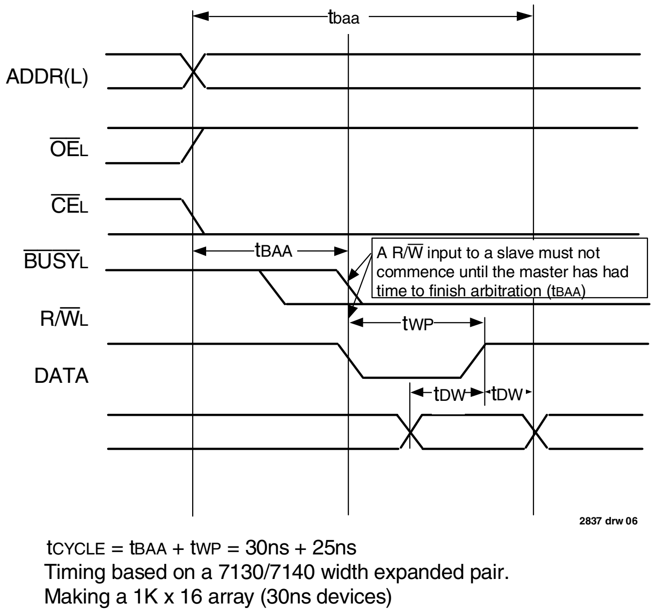
tbaa
ADDR(L)
OEL
CEL
BUSYL
R/WL
DATA
tBAA
A R/W input to a slave must not commence until the master has had time to finish arbitration (tBAA)
tWP
tDW
tDW
2837 drw 06
tCYCLE = tBAA + tWP = 30ns + 25ns
Timing based on a 7130/7140 width expanded pair.
Making a 1K x 16 array (30ns devices)

Figure 6. Width Expansion Timing

Why should I use a dual-port over a SRAM and some logic?

There are actually quite a few considerations that go into this decision and some of these are not so obvious. The following is a list of a few:

a) Performance
b) Improved Bandwidth
c) Logic savings
d) Power savings
e) Reliability improvements
f) Design time

Let's discuss each of these topics individually:

Performance

CPU's must take turns for access to the SRAM, requiring two cycles plus the delay associated with multiplexing. Access to the dual-port is simultaneous, effectively doubling the bandwidth.

A multiplexed SRAM must operate at 3ns to acheive bandwidth equivalent to a 10ns dual-port.

Logic savings

The logic savings is relatively straightforward. If the block diagram of the dual port RAM is duplicated the design would require some form of comparator, which could get rather large if it were to compare every address. At a minimum some form of arbitration logic is required. There are many schemes to this with some of the more basic approaches being handled in a PAL, as with any scheme like this, questions of metastability and settling time must be addressed.

It is also a hidden factor that the cost of a logic solution may look a slight bit less expensive but if you add in the additional board savings, PAL documentation, PAL fusing, parts inventory, and manufacturing cost, you will find a significant savings in designing with a device like a dual port.

Power savings

The actual amount of power savings will be purely dependant on how complex the support logic has to be around the SRAM array. If there is a requirement of full address arbitration and/or any of the other logic functions, such as semaphore logic, it can get quite extensive. For instance, if PALs are used to perform these functions, the higher speed PALs can draw significant current.

Reliability improvements

The reliability (MTBF) of a circuit is dependent on many factors such as circuit complexity, power dissipation, number of pins, maturity of product, and even the P.C. board used. In many of these systems a dual port can be a plus. As mentioned earlier, it is very possible that the complexity of a logic implementation of this device can be quite high. This could significantly impact the MTBF of your design.

Upgrade path

As mentioned earlier, many of the dual-port products provide a range of densities and many of them are pin-compatible. This provides an easy upgrade path. In addition multiple speed grades are available which allow for additional circuit margining in the same footprint.

Design time

Here again, it is not intended to say that the design of the logic surrounding a dual-ported memory is incredibly complex, but it can take a significant amount of time to design, simulate, create timing diagrams, PAL equations, do the board layout, and system debug. It can lead to a loss of design time for other more complex portions of your design. As with many functions these days, a greater amount of integration leads to faster, more confident, design times, with less time spent on the standard functions like RAM arrays.

What advantage does a dual-port have over a FIFO?

When the applications of a dual port are examined, they have a very similar set of usages as does a FIFO. It can be used for passage of critical data from one higher speed device to a slower device, it can be used to pass data in multiprocessor environments, etc. But the major advantage of a dual-port is that the data is accessible in a random fashion, and is retrievable if multiple reads of the same data is required. A FIFO allows only sequential access of the data and in most cases the data can be read only once.

Can the part be used in a "pass-through" mode?

The first thing that should be explained is what is "pass-through" mode. This is a situation in which both ports of a dual-port RAM are accessing the same location and one port is performing a write and one a read. This would require the device to be either a slave type device or a master device in which the side performing the write wins the arbitration (so as not to disable the write from occurring internally). The term "pass-through" means that the data being written on one port will ripple through to the other port if the proper amount of time is allotted.

This is a valid access with an IDT dual port and does set us apart from some of our competitors, who actually register the data at the time of the request and therefore another complete access must be initiated to access this new data.

The timing parameter associated with this parameter is the t_{DDD} parameter. There are qualifiers with this parameter as with all timing parameters. The t_{DDD} time is based on the opposite port completing a valid write cycle, which is why the t_{DDD} is based on the start of a t_{DW} (data valid to end of write time). In general,the timing runs about 10ns longer than a standard read access (i.e. for a 25ns t_{AA} device t_{DDD} is 35ns).

Can I use the x16 devices for byte writes and reads?

To answer there are two devices to discuss. The first is the 7133/7143 pair. These are the 2K x 16 devices. These devices have two separate write enable signals (R/$\overline{W}_{UB}$, R/$\overline{W}_{LB}$) on each side but only one chip enable. This makes it easy to perform byte writes but more difficult to perform byte reads. Since there is only one chip enable, a read can only be performed on a 16-bit word, or a write on eight bits and a read on eight bits. To perform byte reads on these devices, it be-comes necessary to place some form of buffer in the data path to allow control over which byte is to be read from the device.

On the larger x16 devices from IDT, starting with the 4K x 16 device (IDT7024), pins were added to allow for both write and read accesses on a byte level. These devices have upper and lower byte signals used in conjunction with the R/$\overline{W}$, $\overline{CE}$ and $\overline{OE}$ signals to allow for byte control. To perform a write, a combination of R/$\overline{W}$(LOW) and $\overline{UB}$ and/or $\overline{LB}$(LOW) signals as well as $\overline{CE}$ must be combined. For a read a combination of R/$\overline{W}$(HIGH) and $\overline{UB}$ and $\overline{LB}$(LOW) as well as $\overline{CE}$ must be combined. This now allows for byte accesses directly out of the dual-port without any additional buffering outside the device. This same philosophy has been carried through to the x18 and x36 devices as well.

Is there any way to prioritize the arbitration scheme of the device?

There is no way to define a prioritized arbitration scheme between asynchronous sides of a Dual-Port. The device will look at address and $\overline{\text{CE}}$ inputs and arbitrate who will get the $\overline{\text{BUSY}}$ signal. The only way to force one side to win the arbitration is to synchronize the two ports' requests skewing one by 5ns or more. The earlier arriving of the two will "win" and the opposite port will receive the $\overline{\text{BUSY}}$.

What applications do you see these devices going into?

As far as IDT is concerned, dual ports should be used in every electronic device known to man. In actuality there are some key areas that we have been strong in and others that just make sense for a device like this. Some examples of these are:

- Cellular Base Stations
- Cellular Phones
- Multi-Protocol Routers
- Internet Appliances
- LAN/WAN Switches
  - ATM
  - Ethernet
  - Fast Ethernet
  - Gigabit Ethernet
- PBXs
- RAIDs
- Set-top Boxes
- Video Conferencing
- Audio Video Editing
- Call Distribution Systems
- Graphics Accelerators
- Ultrasound Imaging
- Satellite Encoders
- Cable Modems
- Aerospace Instrumentation
- Flight Simulators
- Industrial Controls

Any place where data must be passed between devices.

Summary

Many different concerns regarding the use of dual ports have been addressed here. It you have more, please contact your local IDT applications engineer. We hope that these questions have shed some light for you into how the dual ports offered by IDT operate. Good luck designing!

Corporate Headquarters

TOYOSU FORESIA, 3-2-24 Toyosu,
Koto-ku, Tokyo 135-0061, Japan
www.renesas.com

Contact Information

For further information on a product, technology, the most up-to-date version of a document, or your nearest sales office, please visit www.renesas.com/contact-us/.

Trademarks