USING IDT7024 AND IDT7025 DUAL-PORT STATIC RAMs TO MATCH SYSTEM BUS WIDTHS

APPLICATION NOTE AN–59

By Bhanu V. R. Nanduri

Introduction

This application note describes three design approaches to accomplish bus width matching using the IDT7024 and the IDT7025 dual-port static RAMs. Interfacing 32-bit buses to 16-bit buses, 32-bit buses to 8-bit buses and 16-bit buses to 8-bit buses is described in detail. In general, any bus that is a multiple of 8 bits can be efficiently interfaced to any other bus that is also a multiple of 8 bits using these dual-port RAMs.

The IDT7024 (4K x 16) and the IDT7025 (8K x 16) dual-port static RAMs are identical to each other in every respect except depth. For simplicity, only the IDT7024 will be discussed in detail. The IDT7024 and the IDT7025 dual-port static RAMs are provided with left and right upper byte enable ($\overline{UB}_L$ and $\overline{UB}_R$) and the left and right lower byte enable ($\overline{LB}_L$ and $\overline{LB}_R$) inputs. These byte enables allow interfacing in any bus width matching scheme without the need for external tri-state buffers or transceivers.

Bus matching schemes require that the byte ordering of information be maintained. This byte ordering can be either "***big-endian*** " or "***little-endian***". If data is configured in a *big-endian* format, byte 0 is always the leftmost byte. *Big-endian* is predominant in machines such as the MC 68000 and the IBM 370. If data is configured in a *little-endian* format, byte 0 is always the least significant, rightmost byte. *Little-endian* is used in machines such as the Intel x86, NS 32000 and the DEC VAX. The MIPS R3000 microprocessor and the Intergraph CLIPPER support both data formats, however both these machines must be informed at "power on reset" which data format will be used. The *big-endian* and the *little-endian* byte ordering format is pertinent to 16-bit, 32-bit and 64-bit machines and is not applicable to 8-bit machines. Figures 1a to 1d illustrate the possible *big-endian* and the *little-endian* data conversions.

This discussion on interfacing buses takes into account the byte

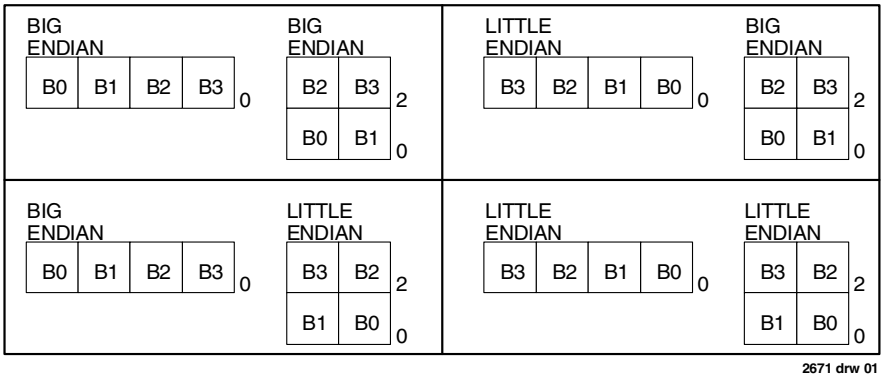

Figure 1a. *Little-endian* and *Big-endian* Byte Mapping Between 32-bit and 16-bit Buses — B_0, B_1, B_2 and B_3 are Bytes Within the 32-bit and 16-bit Words

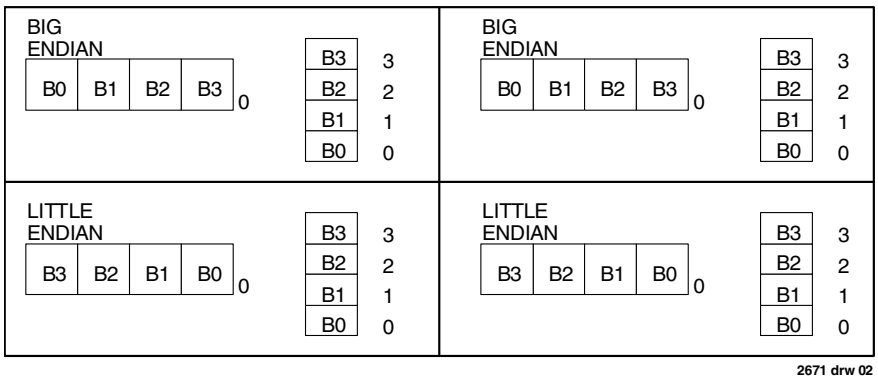

Figure 1b. *Little-endian* and *Big-endian* Byte Mapping Between 32-bit and 8-bit Buses — B_0, B_1, B_2 and B_3 are Bytes Within the 32-bit and 8-bit Words

MARCH 2000

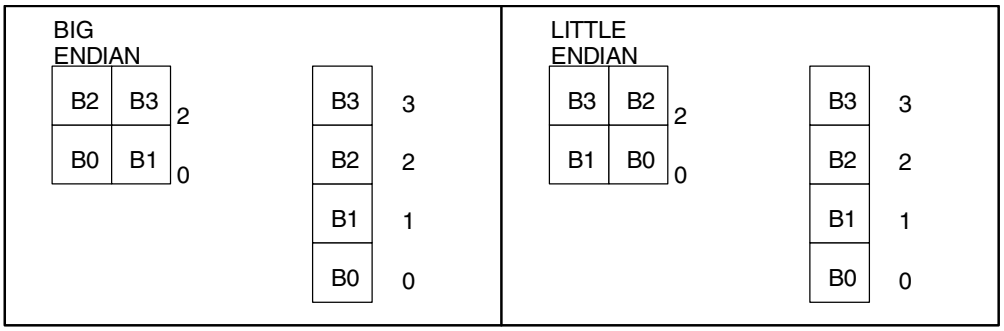

Figure 1c. *Little-endian* and *Big-endian* Byte Mapping Between 16-bit and 8-bit Buses — B_0, B_1, B_2 and B_3 are Bytes Within the 16-bit and 8-bit Words

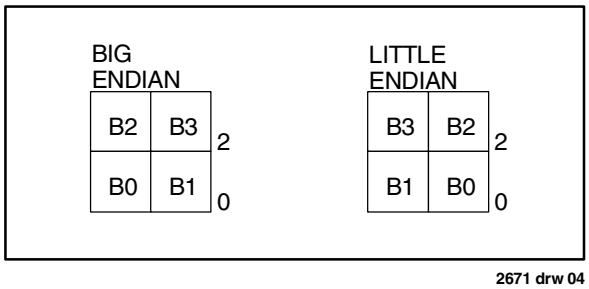

Figure 1d. *Little-endian* and *Big-endian* Byte Mapping Between 16-bit Buses — B_0, B_1, B_2 and B_3 are Bytes Within the 16-bit Words

ordering of data using either the *big-endian* or the *little-endian* data format and even shows how to share data between a big-endian and a little-endian system. This is included to serve as a guide only and is in no way exhaustive. The user is urged to investigate further the data organization to be used in his or her design before attempting to interface buses using dual-port SRAMs.

Figure 2 shows a 16-bit bus to an 8-bit bus interface where the 16-bit side is assumed to be using the *little-endian* data format. On the 8-bit side of the interface, high order data lines D_{15} - D_8 on the SRAM are connected to low order data lines D_7 - D_0 of the SRAM (D_{15} to D_7, D_{14} to D_6, etc.)

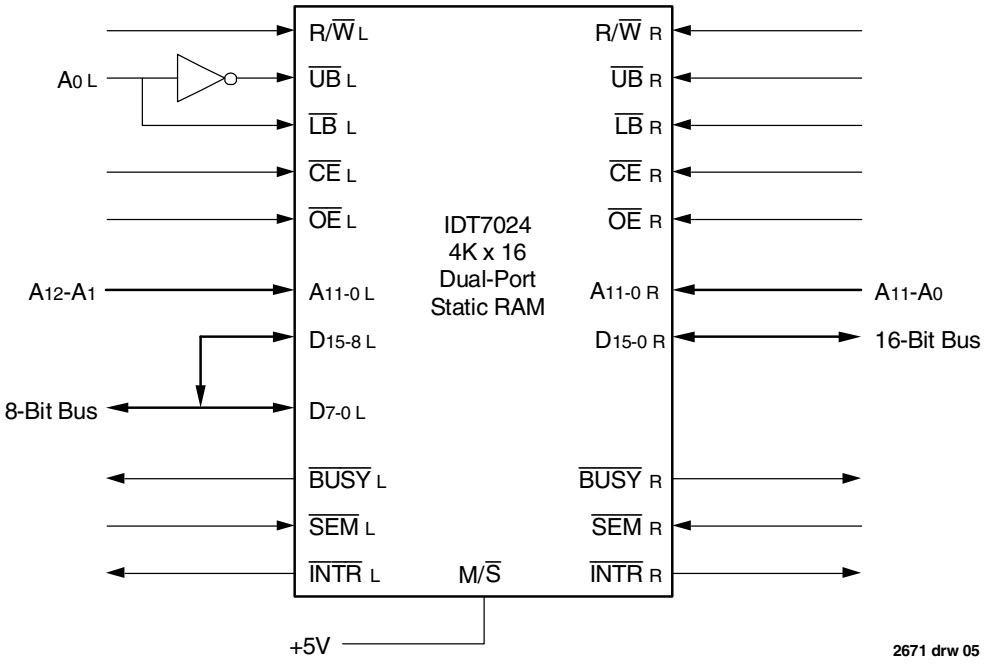

Figure 2. An Interface to Connect 8- and 16-Bit Buses

and processor address line A_0 is used to select the lower or higher order byte. When A_0 is V_{IL}, SRAM byte 0 is selected and when A_0 is V_{IH}, SRAM byte 1 is selected. Address lines A_{12} - A_1 of the 8-bit processor are connected to the twelve address lines of the IDT7024, as shown. These address lines are used to select the 4K words of the IDT7024. If the 16-bit bus side uses the *big-endian* data format instead of the *little-endian* data format then, on the 8-bit side, A_0 must be a V_{IH} to select byte 0 and A_0 must be a V_{IL} to select byte 1 to guarantee correct byte ordering on the 8-bit side. An alternate approach to ensure correct byte ordering on the 8-bit side is to place the inverter shown in Figure 2 on the $\overline{LB}_L$ line instead of the $\overline{UB}_L$ line. This change will ensure that when the 8-bit side's A_0 line is V_{IL}, RAM byte 1 will be selected and, when the A_0 line is V_{IH}, SRAM byte 0 will be selected.

Figure 3 is an interface that connects a 32-bit bus to a 16-bit bus. In Figure 3 both the 32-bit side and the 16-bit side are assumed to be *little-endian*. The upper chip in the diagram holds the two lower order bytes of the 32-bit word and the lower chip in the diagram holds the two higher order bytes. In this interface, processor address bit A_0 on the 16-bit side is used to select a 16-bit SRAM word. When A_0 is V_{IL} the SRAM's lower order sixteen bits are selected and, when A_0 is V_{IH}, the higher order sixteen bits are selected. Selection of the upper byte or the lower byte of either of the two SRAMs is determined by the upper byte enable ($\overline{UB}$) and the lower byte enable ($\overline{LB}$) inputs. If a *big-endian* byte ordering is assumed on both the 32- and the 16-bit sides, the $\overline{UB}$ and the $\overline{LB}$ inputs to the IDT74FCT139 must be interchanged on the 16-bit side.

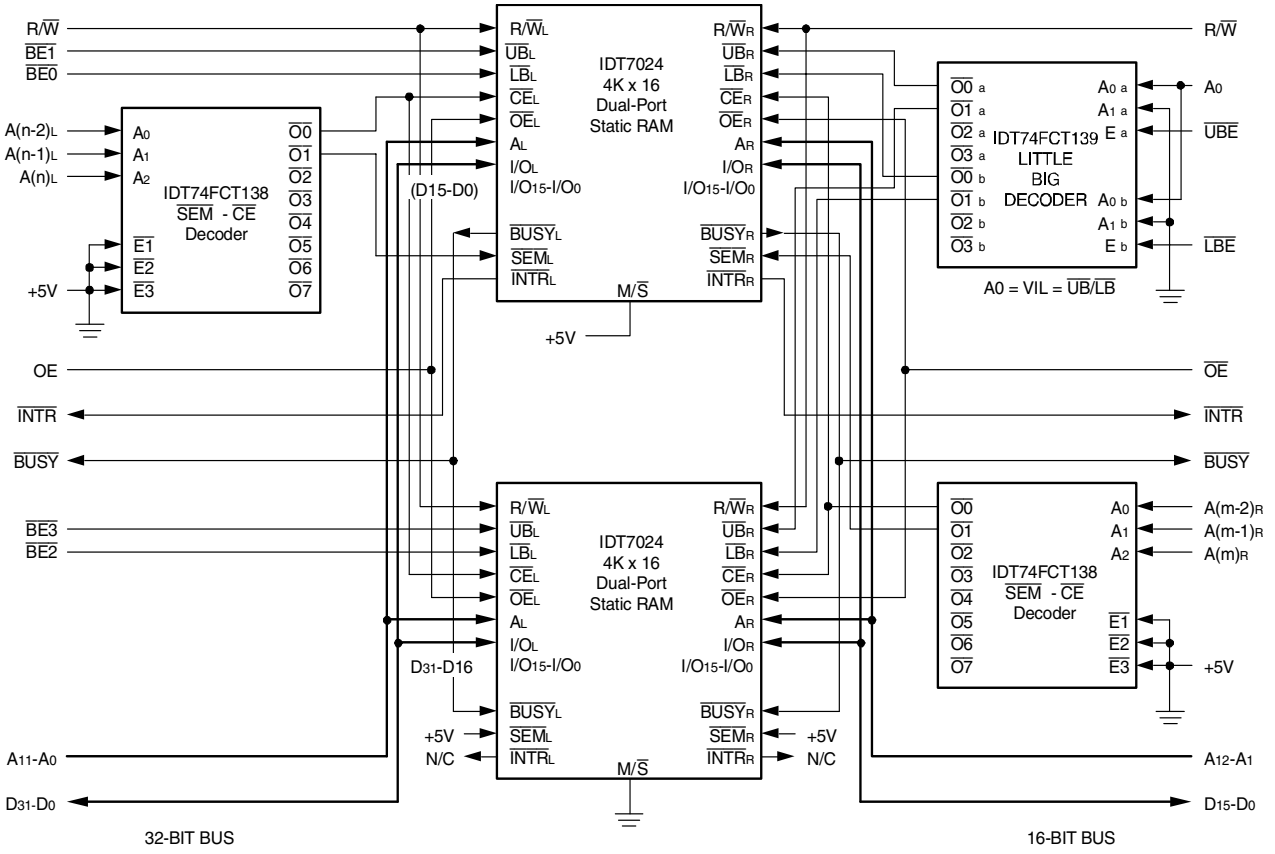

Figure 3. An Interface to Connect a 32-Bit Bus to a 16-Bit Bus

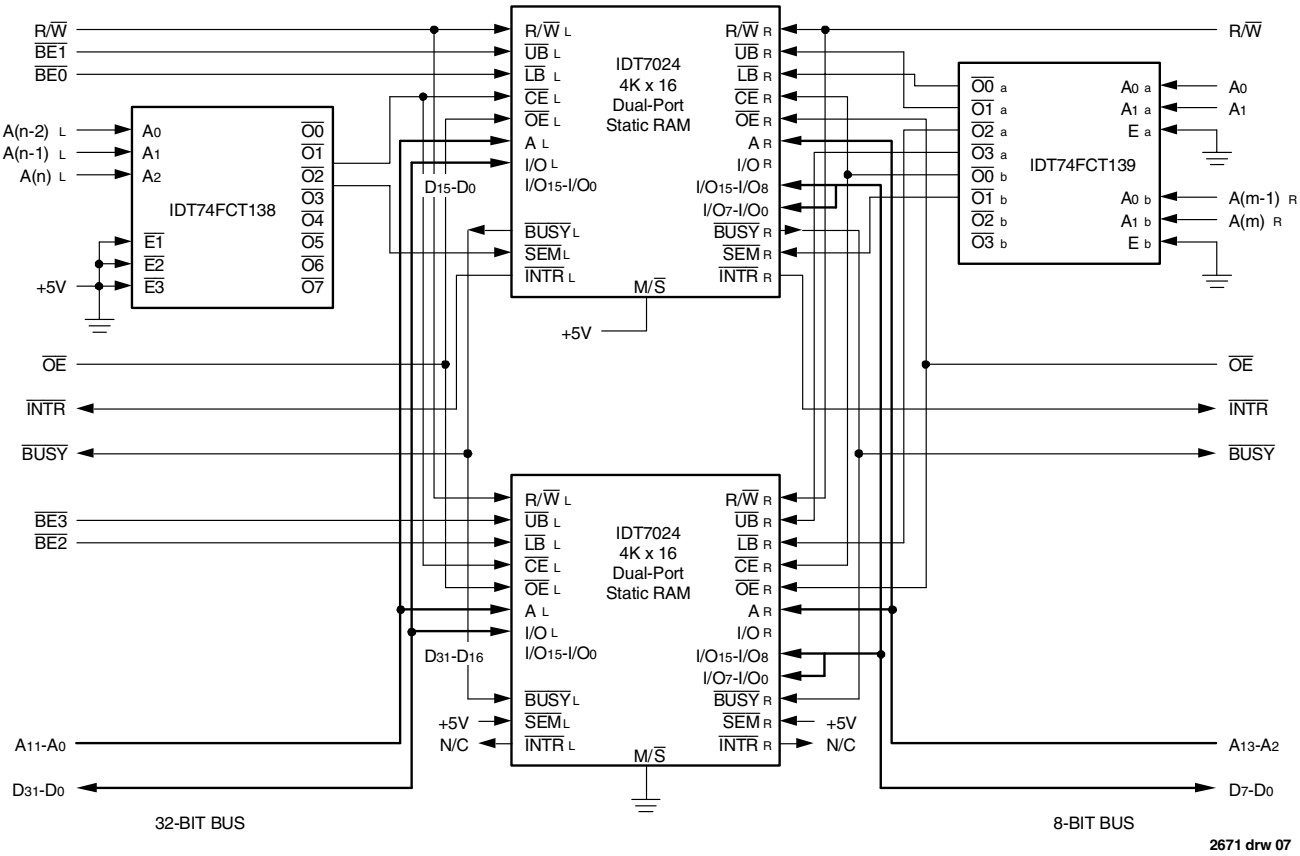

Figure 4. An Interface to Connect a 32-Bit Bus to an 8-Bit Bus

Figure 4 is an interface that connects a 32-bit bus to a 8-bit bus. In Figure 4, the 32-bit side is assumed to be *little-endian* byte ordered. The upper chip in the diagram holds the two lower order bytes and the lower chip in the diagram holds the two higher order bytes. In this interface, address bits A_0 and A_1 on the 8-bit side are used to select an 8-bit word. Table 1 illustrates the bytes selected for each combination of A_0 and A_1 for a *little-endian* byte-ordered data format on the 32-bit side. If a *big-endian* byte-ordering is assumed on the 32-bit side, the upper chip will hold the two higher order bytes and the lower chip holds the two lower order bytes. Address bits A_0 and A_1 on the 8-bit side are used to select one of four bytes as illustrated in Table 2. The mapping scheme to accomplish other bus interfaces is left to the user.

8-bit Side		32-bit Side			
A_0	A_1	$\overline{BE3}$	$\overline{BE2}$	$\overline{BE1}$	$\overline{BE0}$
0	0	X	X	X	0
0	1	X	X	0	X
1	0	X	0	X	X
1	1	0	X	X	X

2671 tbl 01

Table 1. Byte Selection Equivalency Assuming the 32-Bit Side Uses the *Little-Endian* Byte Ordering of Data

8-bit Side		32-bit Side			
A_0	A_1	$\overline{BE0}$	$\overline{BE1}$	$\overline{BE2}$	$\overline{BE3}$
0	0	X	X	X	0
0	1	X	X	0	X
1	0	X	0	X	X
1	1	0	X	X	X

2671 tbl 02

Table 2. Byte Selection Equivalency Assuming the 32-Bit Side Uses the *Big-Endian* Byte Ordering of Data

Busy Arbitration Logic

$\overline{BUSY}$ arbitration is performed only when there is an address match and the chip enables are active. The IDT7024 and the IDT7025 dual-port SRAMs are provided with a master/slave (M/$\overline{S}$) pin through which the user can configure the $\overline{BUSY}$ logic on these devices to operate as masters or slaves. $\overline{BUSY}$ arbitration is performed only by the master, which generates the busy signal. The master outputs a logic "0" on the busy line of the port that loses arbitration, at the same time it generates an internal write inhibit signal to block any write operation on the losing port. When configured to operate as slaves, these devices use the $\overline{BUSY}$ line as an input. The slave takes the $\overline{BUSY}$ line as an input and generates an internal write inhibit on the same port that received the $\overline{BUSY}$. The upper and lower byte enable inputs do not affect the operation of $\overline{BUSY}$ logic in these devices. If $\overline{BUSY}$ logic and width expansion are being used, it is important that the $\overline{CE}$ of the master and the associated slave always be active at the same time. If the decoding logic allows the slave to be selected without the master, the $\overline{BUSY}$ logic will not operate correctly. Care has been taken in both Figures 3 and 4 to assure correct $\overline{BUSY}$ logic operation. It should be kept in mind, however, that $\overline{BUSY}$ logic is often not an essential part of a dual-port SRAM-based system. The user is urged to read Application Note AN-02 for more information on busy logic arbitration.

Interrupt Logic

The IDT7024/IDT7025 dual-port SRAM chips have interrupt generation capability that can be very effectively used to interrupt processors connected to either side of the dual-ports. A processor connected to the left port can generate an interrupt to the processor connected on the right port by writing to the topmost location in the memory array. In the case of the IDT7024, this location is FFF (Hex). The processor on the right port clears the interrupt by reading from this location, i.e. FFF (Hex). Similarly, the processor on the right port can interrupt the processor on the left port by writing to the topmost minus one location, i.e. FFE (Hex), for the IDT7024. The processor on the left port clears the interrupt by reading from location FFE (Hex).

Side	Set Address (HEX) (Write) (Interrupts the Other Side)	Clear Address (HEX) (Read) (Clears the Interrupt on This Side)
Using left port	FFF	FFE
Using right port	FFE	FFF

2671 tbl 03

Table 3. Interrupt Set and Clear Addresses for the IDT7024 Dual-Port RAMs

Side	Set Address (HEX) (Write)	Clear Address (HEX) (Read)
32-Bit side (Using left ports)	FFF	FFE
16-Bit side (Using right ports)	1FFC	1FFE
32-Bit side (Using right ports)	FFE	FFF
16-Bit side (Using left ports)	1FFE	1FFC

2671 tbl 04

Table 4. Interrupt Set and Clear Addresses for the 32-bit to 16-bit Interface Shown in Figure 3

Side	Set Address (HEX) (Write)	Clear Address (HEX) (Read)
32-Bit side (Using left ports)	FFF	FFE
8-Bit side (Using right ports)	7FFA or 7FFB	7FFC or 7FFD
32-Bit side (Using right ports)	FFE	FFF
8-Bit side (Using left ports)	7FFC or 7FFD	7FFA or 7FFB

2671 tbl 05

Table 5. Interrupt Set and Clear Addresses for the 32-bit to 8-bit Interface Shown in Figure 4

Table 3 summarizes the interrupt set and clear addresses for the IDT7024 dual-port RAMs, while Tables 4 and 5 summarize the interrupt set and clear addresses for the interface shown in Figures 3 and 4. In the interface schemes illustrated in Figures 3 and 4, we have two dual-ports that have been used to expand the memory in width. This means that we can have two interrupt lines going active one for each chip. The schemes illustrated in Figures 3 and 4 show only interrupts from the master chip being used by either side, while the interrupts from the slave chip are not used.

Semaphore Arbitration

The IDT7024 and IDT7025 are provided with semaphore logic in the form of eight dual-port semaphore flags that are independent of the memory array. These eight cells can be used to supervise the accesses to a maximum of eight blocks of memory. There is no hardware interaction between the semaphores and the SRAM. Address bits $A(m)_R$, $A(m-1)_R$ and $A(m-2)_R$ in Figure 3 are inputs to an IDT74FCT138 which decodes the dual-port SRAM space and the semaphore space on the 16-bit side. Address bits $A(m)_R$ and $A(m-1)_R$ in Figure 4 are inputs to an IDT74FCT139 which decodes the dual-port SRAM space and the semaphore space on the 8-bit side. Similarly, address bits $A(n)_L$, $A(n-1)_L$ and $A(n-2)_L$ are inputs to another IDT74FCT138 which decodes the various address spaces on the 32-bit side. It is necessary to keep the dual-port memory space and the semaphore address space separate. Operating on the two spaces is a mutually exclusive operation and, therefore, chip enable ($\overline{CE}$) and the semaphore enable ($\overline{SEM}$) inputs must never be active at the same time. The semaphore cells are intended to assist software-based protocols intended to prevent address collisions.

The IDT semaphore cells are designed to be used in a clear-and-test sequence. Each cell is normally in the "1" state, indicating that neither side has been assigned the associated block of memory ("No grant"). To access a particular block of memory, one must perform the clear-and-test sequence necessary to get a "grant" from the semaphore cell representing the block. To get a "grant", one must select the semaphore cell representing the associated block of memory, write a "0" (request) to the semaphore cell and then read (test) the semaphore cell to see if a "0" was put out by the cell.

A semaphore cell is selected by asserting the semaphore enable line ($\overline{SEM}$) and by selecting one of the eight semaphore cells with the help of the three lower most address lines A_2 - A_0. In the read operation, if the

Semaphore Address			Selected Semaphore Cell
A_2	A_1	A_0	
0	0	0	Sem Flag 0
0	0	1	Sem Flag 1
0	1	0	Sem Flag 2
0	1	1	Sem Flag 3
1	0	0	Sem Flag 4
1	0	1	Sem Flag 5
1	1	0	Sem Flag 6
1	1	1	Sem Flag 7

2671 tbl 06

Table 6. Semaphore Address Map for the 32-bit Side in Figures 3 and 4

Semaphore Address				Selected Semaphore Cell
A_3	A_2	A_1	A_0	
0	0	0	X	Sem Flag 0
0	0	1	X	Sem Flag 1
0	1	0	X	Sem Flag 2
0	1	1	X	Sem Flag 3
1	0	0	X	Sem Flag 4
1	0	1	X	Sem Flag 5
1	1	0	X	Sem Flag 6
1	1	1	X	Sem Flag 7

2671 tbl 07

Table 7. Semaphore Address Map for the 16-bit Side in Figures 3

Semaphore Address					Selected Semaphore Cell
A_4	A_3	A_2	A_1	A_0	
0	0	0	X	X	Sem Flag 0
0	0	1	X	X	Sem Flag 1
0	1	0	X	X	Sem Flag 2
0	1	1	X	X	Sem Flag 3
1	0	0	X	X	Sem Flag 4
1	0	1	X	X	Sem Flag 5
1	1	0	X	X	Sem Flag 6
1	1	1	X	X	Sem Flag 7

2671 tbl 08

Table 8. Semaphore Address Map for the 8-bit Side in Figures 4

semaphore cell is a "0", that particular block of memory is "available" for use by the side requesting access. If the semaphore cell is a "1", the side requesting access has a "no grant" and that particular block of memory is in use by the other side. In the IDT7024 and IDT7025, the semaphore cells broadcast the "grant" or "no grant" condition on the entire sixteen bits of the data pins. The status of the upper and lower byte enables has no effect on the semaphore request operation.

Consistent with the rest of our discussion on busy logic and interrupts, we will consider the semaphore flags in the IDT7024 containing the lower order sixteen bits of data (master). When the 32-bit side in Figures 3 and 4 reads the semaphores, the processor on the 32-bit side will look at the lower sixteen bits only to check for a "grant" or a "no grant" condition. The 16-bit and 8-bit sides access the semaphore space in Figures 3 and 4 and must read the master IDT7024 containing only the lower sixteen bits to check for a "grant" or a "no grant" condition. (Refer to Tables 6, 7 and 8.)

Summary

Interfacing various buses with the help of dual-ports can be implemented very easily and with a minimum of components. Byte reordering can also be accommodated easily. IDT7024 and IDT7025 dual-port static RAMs have built-in arbitration schemes, upper and lower byte enables, and pin selectable master/slave functions. These features have been designed to aid system designers in their quest for compact, simple and more reliable designs.

IMPORTANT NOTICE AND DISCLAIMER

RENESAS ELECTRONICS CORPORATION AND ITS SUBSIDIARIES (“RENESAS”) PROVIDES TECHNICAL SPECIFICATIONS AND RELIABILITY DATA (INCLUDING DATASHEETS), DESIGN RESOURCES (INCLUDING REFERENCE DESIGNS), APPLICATION OR OTHER DESIGN ADVICE, WEB TOOLS, SAFETY INFORMATION, AND OTHER RESOURCES “AS IS” AND WITH ALL FAULTS, AND DISCLAIMS ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, ANY IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT OF THIRD-PARTY INTELLECTUAL PROPERTY RIGHTS.

These resources are intended for developers who are designing with Renesas products. You are solely responsible for (1) selecting the appropriate products for your application, (2) designing, validating, and testing your application, and (3) ensuring your application meets applicable standards, and any other safety, security, or other requirements. These resources are subject to change without notice. Renesas grants you permission to use these resources only to develop an application that uses Renesas products. Other reproduction or use of these resources is strictly prohibited. No license is granted to any other Renesas intellectual property or to any third-party intellectual property. Renesas disclaims responsibility for, and you will fully indemnify Renesas and its representatives against, any claims, damages, costs, losses, or liabilities arising from your use of these resources. Renesas' products are provided only subject to Renesas' Terms and Conditions of Sale or other applicable terms agreed to in writing. No use of any Renesas resources expands or otherwise alters any applicable warranties or warranty disclaimers for these products.

(Disclaimer Rev.1.01 Jan 2024)

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

Renesas and the Renesas logo are trademarks of Renesas Electronics Corporation. All trademarks and registered trademarks are the property of their respective owners.