Internetworking the SN74ABT3614

Chris Wellheuser and Vasanta Madduri
Advanced System Logic – Semiconductor Group

SCAA018A
March 1996

IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain applications using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

Contents

List of Illustrations

Introduction

Texas Instruments (TI's) SN74ABT3614 is a clocked first-in, -first-out (FIFO) memory with enhanced features required by today's complex networks. The FIFO has all the necessary logic for performing bus-matching, byte-swapping, parity-generation, and parity-checking functions integrated onto a single chip. This on-chip integration greatly simplifies the system designer's job in several ways:

- Easier interconnection of complex networks
- Reduces system bottlenecks
- Helps meet critical timing delays
- Reduces board area

This application report describes the architectural implementation of each of these enhanced features. The first section provides an overview of the SN74ABT3614 architecture and the enhanced features and applications rationale for combining these features with FIFO memory. The second section describes port-B configurations for bus matching and byte swapping. The second section highlights the internal register arrangements, the order of their access during port-B configurations, and the effect of these configurations on the flag operation. The third section describes parity-generation and parity-checking schemes. The fourth section includes an internetworking application example of the SN74ABT3614.

FIFO Architecture

The SN74ABT3614 functional block diagram is shown in Figure 1. The device utilizes dual-port SRAM architecture to provide simultaneous read and write access. Port A is a 36-bit-wide port and port B can be programmed to variable bus widths. All data transfers through a port are synchronized to the low-to-high transition of a continuous port clock by enable signals. The port clocks are independent of one another and can be asynchronous or coincident.

Two independent 64×36 FIFOs (FIFO1 and FIFO2) provide bidirectional data buffering between the two ports. Each of these FIFOs supports clock frequencies up to 67 MHz and has data access times as fast as 10 ns. Both FIFOs have full (FF), almost-full ($\overline{AF}$), almost-empty ($\overline{AE}$), and empty ($\overline{EF}$) flags to indicate their relative status. The $\overline{AF}$ and $\overline{AE}$ flags are programmable, which provides flexibility and control during data transfers. The two 36-bit mailbox registers, (MAIL1 and MAIL2) provide a path around these FIFOs to transmit information in each direction. Each mailbox register has a status flag to alert the user that data is present in the mailbox.

The bidirectional-FIFO core previously described is coupled with additional logic to perform bus matching, byte swapping, and parity generation and checking. The bus-matching logic enables port B of the chip to perform byte-size, word-size, or long-word-size data transfers. In addition, this logic allows the user to chose big-endian or little-endian configurations with byte (9-bit), word-size (18-bit), and long-word-size (36-bit) implementations. These size implementations can be achieved dynamically. The SN74ABT3614 can be used to facilitate data communication between processors or buses of different widths and speeds. Figure 2 shows an example of a 9-bit HDLC communications controller interfacing to a 36-bit high-speed bus using the SN74ABT3614.

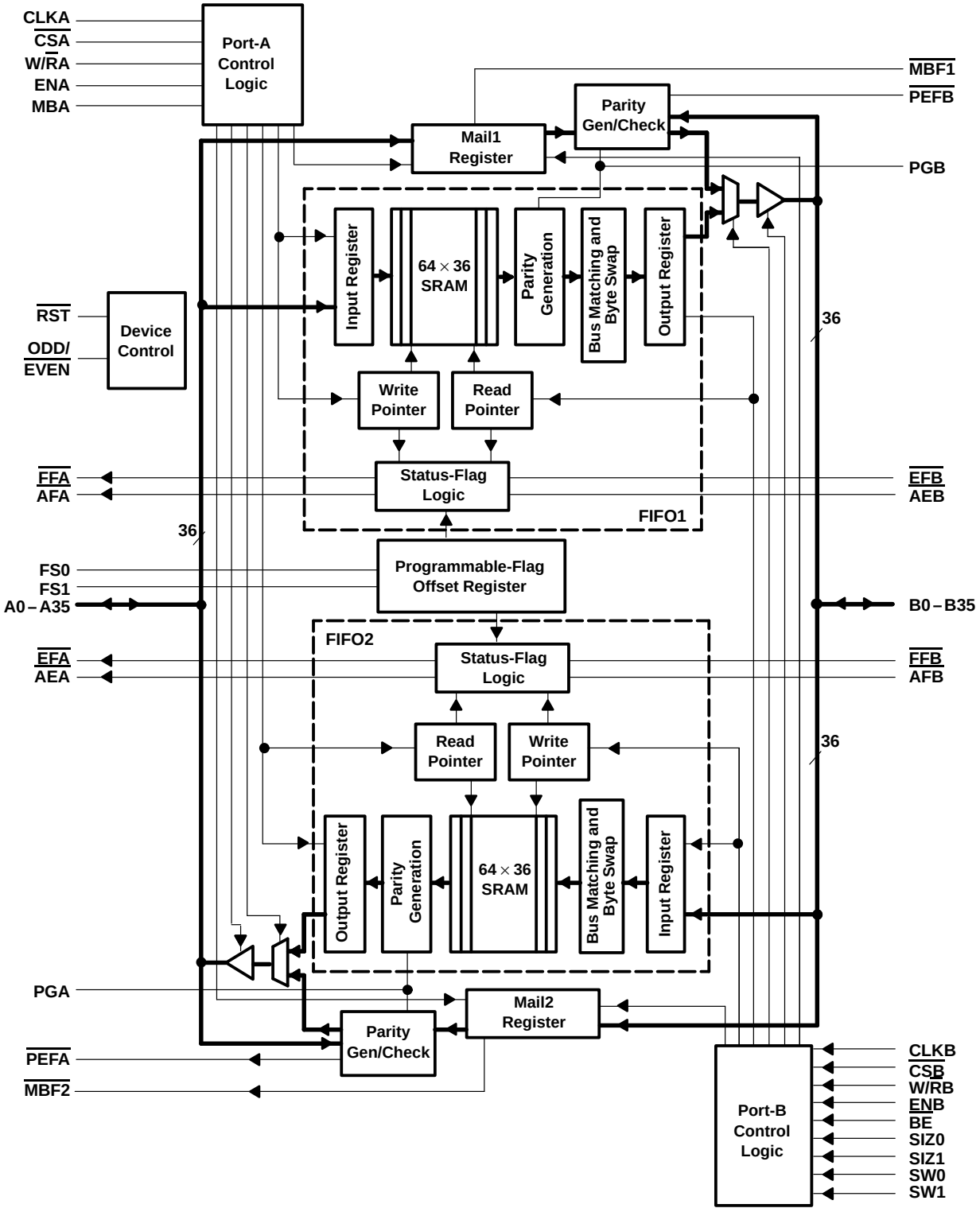


Figure 1. SN74ABT3614 Functional Block Diagram

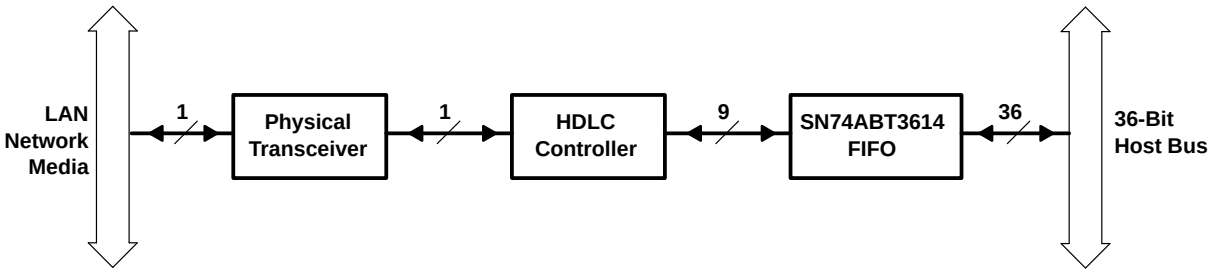


Figure 2. Bus-Matching Example

The byte swapping logic implements four different byte-order arrangements on port B: no swap, byte swap, word swap, and byte-word swap. In particular, the byte-swap operation is useful when a compatible interface between processors from different families is required. For example, the memory organization for the Intel™ 8086 and Motorola 68000 processors is shown in Figure 3. The Intel 8086 uses the little-endian format and Motorola 68000 uses the big-endian format for byte ordering. The byte order within the word has to be swapped in order to establish communication between the two processors. The byte-swap operation of the SN74ABT3614 implements a hardware solution that is faster than a software solution. The byte-swap arrangements can be implemented in conjunction with the size configurations, giving the user great flexibility in configuring networks.

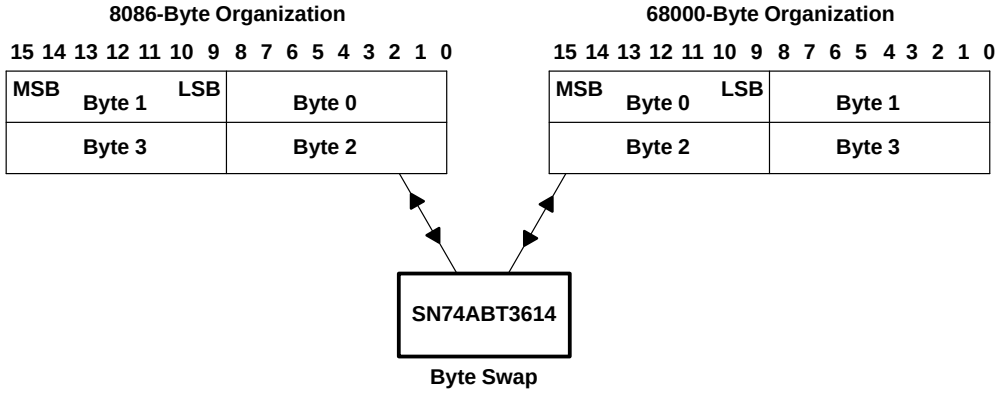


Figure 3. Memory Organization in Different Processor Families

The parity-checking logic allows a form of error checking in data-transmission circuits. Either odd- or even-parity checking can be chosen on incoming data on both port A and port B. A parity error on one or more valid data bytes on a port is alerted by the port's parity-error flag.

Like parity checking, either odd- or even-parity generation can be performed on both ports. The parity-generation logic replaces the most significant bit of each byte to make the total number of ones in the byte (including the parity bit) either odd or even.

Port-B Configuration for Bus Matching and Byte Swapping

Bus Sizing

Port B can be configured for byte-size (9-bit), word-size (18-bit), or long-word-size (36-bit) data transfers, with a choice of big-endian or little-endian formats for byte- and word-size configurations. The size (SIZ1and SIZ0) and big-endian ($\overline{BE}$) input terminals are used to achieve these bus-matching configurations.

During bus-matching operations, each low-to-high transition of the port-B clock (CLKB) cycle stores the levels on the SIZ0, SIZ1, and $\overline{BE}$ input terminals and implements the size on the next clock cycle. Table 1 shows the levels on the five input terminals and the respective size selected. Figure 4 illustrates the data transfer during byte- and word-size configurations. The data transfer for big-endian implementation is indicated by the solid lines; little-endian implementation is indicated by the dotted lines.

Intel is a trademark of Intel Corp.

Table 1. Control of Port-B Bus-Matching Configurations Using SIZ0, SIZ1, and $\overline{BE}$ Terminals

$\overline{BE}$	SIZ1	SIZ0	SIZE OPERATION	B35–B27	B26–B18	B17–B9	B8–B0
X	L	L	Long-word size	Data valid	Data valid	Data valid	Data valid
L	L	H	Word size, big endian	Data valid	Data valid	Invalid	Invalid
H	L	H	Word, little endian	Invalid	Invalid	Data valid	Data valid
L	H	L	Byte, big endian	Data valid	Invalid	Invalid	Invalid
H	H	L	Byte, little endian	Invalid	Invalid	Invalid	Data valid

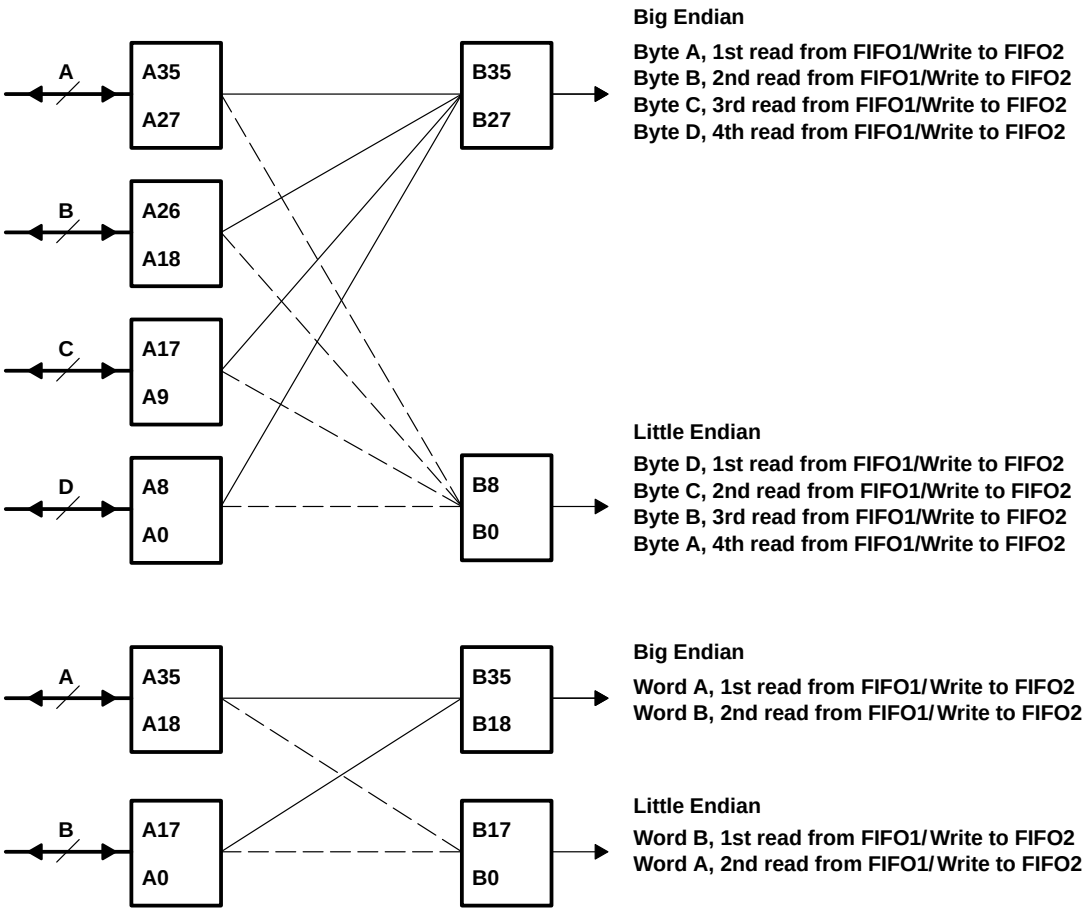


Figure 4. Port-B Bus-Matching Configurations

The bus size can be reconfigured dynamically and synchronous to CLKB rising edge using the SIZ0, SIZ1, and $\overline{BE}$ inputs as control. However, if the low-to-high CLKB transition of a particular CLKB cycle stores new levels on the control terminals, it is the next CLKB cycle that implements the size configuration chosen by those new levels; therefore, bus size can be changed on the fly with a minimum of one clock-cycle latency. If the dynamic bus-sizing option is not exercised, the SIZ0, SIZ1, and $\overline{BE}$ control terminals can simply be hardwired to required levels to implement the port-B size selection.

The bus-matching operation is always handled in auxiliary registers, either after the data is read from FIFO1 SRAM during a port-B read cycle or before the data is written to the FIFO2 SRAM during a port-B write cycle. The following is an in-depth look at read and write accesses on port B during bus-sizing operation and the effect of sizing on the flag operation.

Read Accesses on Port B

To explain how bus sizing is implemented when performing reads from FIFO1 to port B, it helps to start with the internal memory-access architecture of a normally configured 36-bit port. Read accesses are made alternately on odd and even memory

blocks as shown in Figure 5; the interleaved architecture being necessary for high-frequency operation. As each 36-bit word is accessed, it is stored temporarily in one of the two banks of transparent latches represented by TL1 and TL2 in Figure 5. These are multiplexed together and loaded alternately into the output storage registers upon each successive clock pulse. All memory accesses are full-length 36-bit words and each valid read clock produces a full 36-bit word at the outputs.

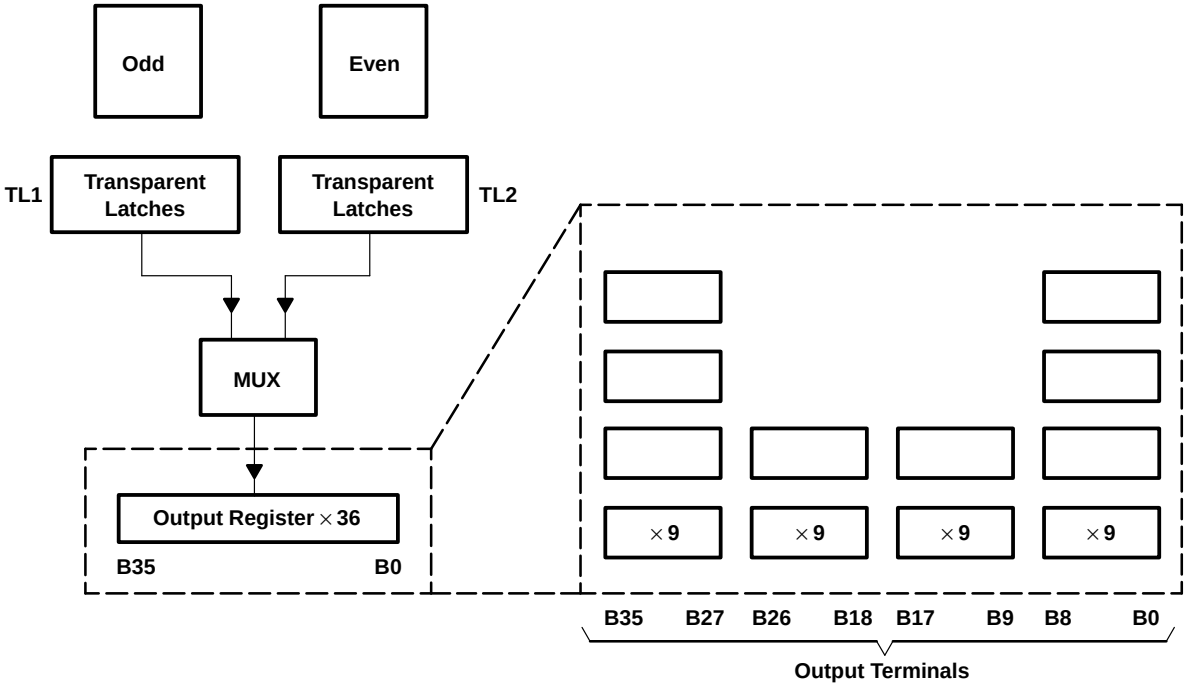
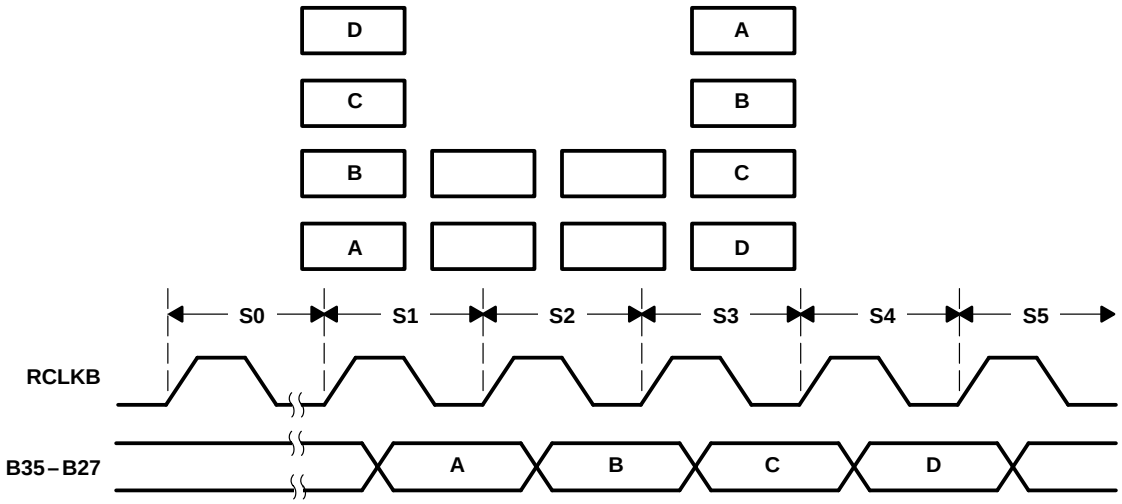


Figure 5. Pipeline Registers

Regardless of bus size, bus swap, or port selection, all internal memory-access operations are based on full 36-bit words and the associated internal flag status also reflects that basis.

To implement bus sizing, the single bank of output storage registers is replaced with multiple queues of 9-bit pipeline registers as shown in Figure 5. When a read access is performed, a full 36-bit word is loaded in groups of nine into these pipeline registers. The location of each byte is specifically loaded and controlled by the state of the port configuration (SIZ0, SIZ1, SW1, SW0, and $\overline{BE}$) terminals. Depending on the size selected, successive read-clock pulses shift out the remaining bytes in each pipeline register. During this time, internal reads from the FIFO1 memory are disabled until the last word or byte is clocked out, which is implemented by a small programmable counter. The counter counts four clock cycles in byte-size mode and two clock pulses when word size is selected. The counter is disabled for normal long-word operation. Figures 6 and 7 show how bus sizing works in practice for read operations on port B.

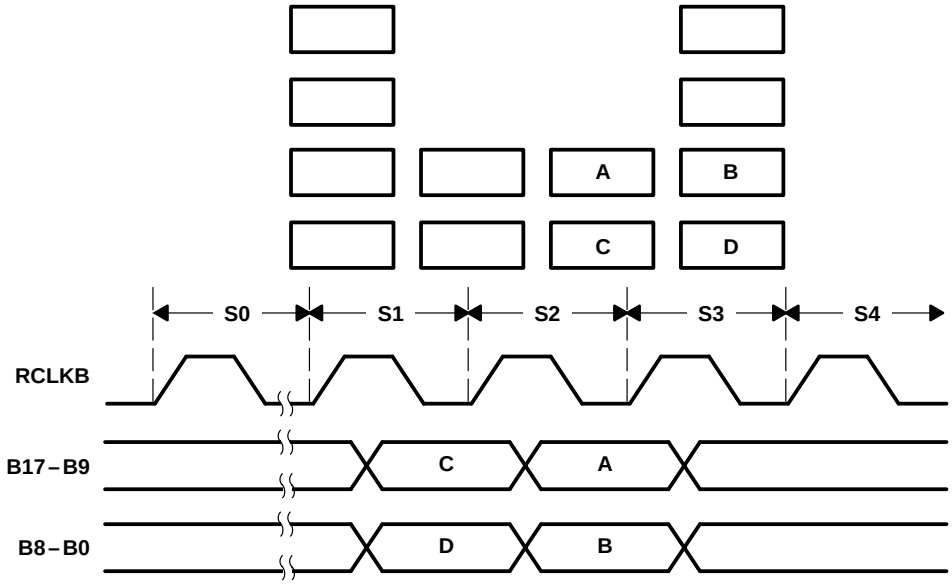
In Figure 6, the bus size that is set up one clock cycle ahead is chosen to be byte length (9 bit). A 36-bit word is accessed from FIFO1 memory and loaded into the pipeline registers in 9-bit bytes, represented by A, B, C, and D. The byte loading shown in Figure 6 works for both big- and little-endian format but only the port chosen transmits data. Assuming the big-endian configuration is chosen, the A byte is present on B35–B27 output terminals after the first read clock. The next three read clocks shift out the remaining bytes in proper sequence; B, C, then D. This sequence is determined on the first read-clock pulse when the full-length word is accessed and loaded into the pipeline registers. The timing diagram shown in Figure 6 describes the internal memory-access and shift-register timing relationships.



S0 – Select byte-size big endian
S1 – Access-memory, load-pipeline queues shown above. A byte is present on outputs.
S2 – Shift B byte to outputs; memory access disabled
S3, S4 – Shift C and D bytes to outputs
S5 – Memory access enabled; load new word from memory into pipeline

Figure 6. Read Access During Byte-Size, Big-Endian Configuration

In Figure 7, the bus size is chosen to be word length with little-endian configuration. The first read clock performs a memory access and loads the 36-bit word into the pipeline registers as shown. Bytes C and D are present on the output terminals B17–B0. The next read clock shifts out bytes A and B, while internal memory access is disabled. As it only requires two clock pulses to access a full 36-bit word in this mode, the internal memory access is reenabled after the second CLKB pulse.



S0 – Select word-size little endian
S1 – Access-memory, load-pipeline queues as shown above. C and D bytes appear on outputs.
S2 – Shift A and B bytes to outputs; memory access disabled
S3 – Memory access enabled; load new word from memory into pipeline

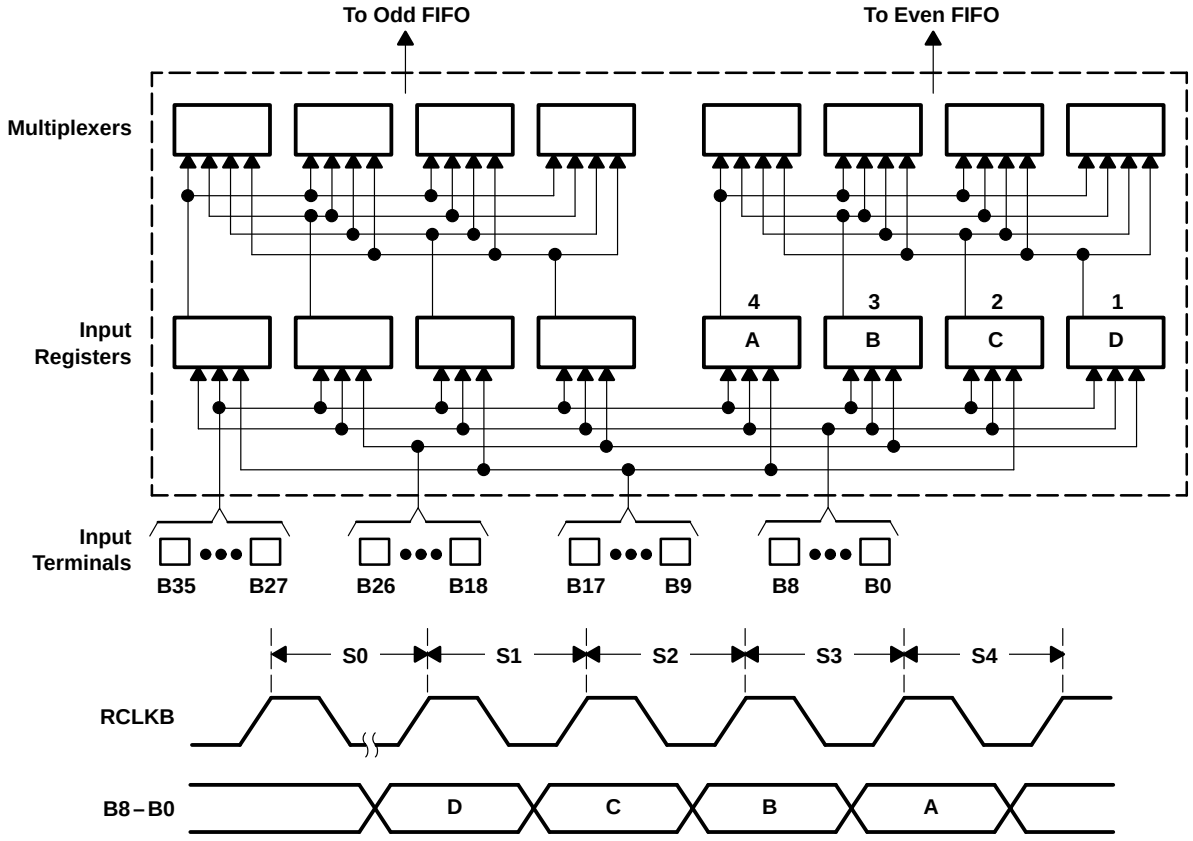
Figure 7. Read Access During Word-Size, Little-Endian Configuration

Write Operation to Port B

When writing to FIFO2, data at the input terminals is stored in one of two sets of input registers on the rising edge of CLKB and transferred into either the odd or even FIFO2 memory array. To accommodate varying bus sizes, each set is divided into four groups of nine registers. Each group of nine can accept input data from any of the nine input terminals as shown in Figure 8. Upon the rising edge of CLKB, data from the selected group of input pins is steered to the appropriate group of nine registers and stored. The group of input pads used and the location in which the successive bytes are stored in the registers are controlled by the state of the port configuration (SIZ0, SIZ1, and $\overline{BE}$) terminals.

Depending on the size selected, either one, two, or four valid CLKB pulses are needed to load a full 36-bit word to memory. Only when the last byte/word is loaded into its group(s) of registers is the full 36-bit word written to the FIFO2 memory array. Until that time, writes to the memory are disabled. A separate counter, similar to the one used for read operations, keeps track of the number of bytes loaded and when to write to the array.

The following example shows how this works in practice: Byte-size and little-endian configurations are selected. Input pads B0–B8 accept data written to FIFO2. Assuming a byte order from MSB to LSB of A, B, C, and D for the full 36-bit word to be written to FIFO2, the bytes need to appear in succession on the active input terminals in the order of D, C, B, and A. Figure 8 shows the position in the groups of registers that each byte occupies when loaded. On the fourth CLKB pulse, which loads the A byte, the data in the four groups of registers is written into FIFO2 memory array.



S0 – Select byte-size little endian
S1 – Write byte D at position 1; writes to FIFO disabled
S2 – Write byte C at position 2; writes to FIFO disabled
S3 – Write byte B at position 3; writes to FIFO disabled
S4 – Write byte A at position 4; load full 36-bit long word to memory

Figure 8. Input Registers and Timing for Write Operation During Byte-Size, Little-Endian Configuration

Effect on Status Flags

Bus sizing affects the operation of the almost-empty ($\overline{AEB}$) and almost-full ($\overline{AFB}$) flags associated with port B and, to some degree, the empty ($\overline{EFB}$) and full ($\overline{FFB}$) flags. The almost-empty and almost-full flags can be programmed to one of the preset values of 4, 8, 12, or 16 during device reset. These depths are based on full 36-bit words. When used with bus sizing, the internal flag still reacts to the programmed depth to a multiple of the byte or word size selected. For example, if a depth of eight long words is programmed and byte sizing is selected, the $\overline{AEB}$ flag indicates when there are 32 bytes remaining to be read from FIFO1 before it goes empty. This results from the internal flag reacting to eight long words remaining in the memory. Due to the byte size being selected, it takes 32 read-clock pulses to unload them. Similarly, if a depth of four long words is programmed and a bus size of 18-bit words is selected, the $\overline{AFB}$ flag reacts when there are eight more valid CLKB pulses before FIFO2 is written full. This is based on the need for two write-clock pulses to load each long word into memory and the internal flag reacting when there are four available locations remaining in the memory.

The empty and full flags for port B still indicate when the associated boundary condition is met during sizing operations because the internal flags for FIFO1 and FIFO2 are based on 36-bit words only. The write to FIFO2 memory during bus-size mode occurs on the last byte or word of the full 36-bit word being loaded. This means $\overline{FFB}$ does not indicate a full condition until all four 9-bit bytes (or both 18-bit words) of the last full long word are clocked into FIFO2. If, for instance, only three of the bytes are loaded during byte sizing, the internal write to memory has not occurred yet, and the status flag does not indicate full until the last byte is clocked in and the internal write to FIFO2 memory takes place.

Likewise, due to an internal long-word read access being performed at the beginning of a bus-sized byte or word transfer, the internal empty-flag status updates immediately, since FIFO1 has effectively been read empty. However, in the instance of byte sizing, there are still three bytes remaining in the pipeline to be shifted out before the user considers the $\overline{EFB}$ flag to be valid. To provide a proper empty-flag indication, the same signal that disables the internal memory accesses during remaining byte (or word) transfer is combined with the synchronized empty-flag output signal and prevents it from switching until the final byte (or word) is clocked to the outputs. This results in the $\overline{EFB}$ flag correctly indicating when the empty-boundary condition occurs, regardless of the bus size chosen.

Dynamically Changing the Bus Size

Many applications select a bus size at system initialization, which remains fixed due to the architectural design of the system. However, applications that require dynamic bus sizing to be performed can use the SN74ABT3614 to provide an effective interface. To avoid data loss when making the transition, the following explanation is provided. Mailbox operations, which are selected with bus-size control terminals, also are discussed.

Bus-size selections must be made one cycle before they are to take effect. The values present on SIZ0 and SIZ1 are stored in registers XFF1 and XFF2 on the rising edge of CLKB as shown in Figure 9. This latency allows the control logic to be set up for the next transfer, resulting in short setup and hold times for these inputs and allows comparisons to the previous state of the inputs to determine if an actual change in bus size is being requested. This is important when using the size inputs to enter mailbox mode, then returning to the previous size for FIFO operations.

In Figure 9, exclusive NOR (XNOR1 and XNOR2) gates are used to compare the current state of the size inputs to their previous states. If either input is changed from its previous state, the signal COUNT_RES is driven low. This signal is a synchronous reset to the counters that keep track of the byte or word count. When a size change is requested, the rising clock edge that loads the new size value into the registers also resets the counters, ensuring proper byte tracking of the new size beginning with the next clock pulse. If a size change is requested before the last byte/word is transferred from/to the FIFO, the remaining data is lost or the entire word is not written (see Figure 10).

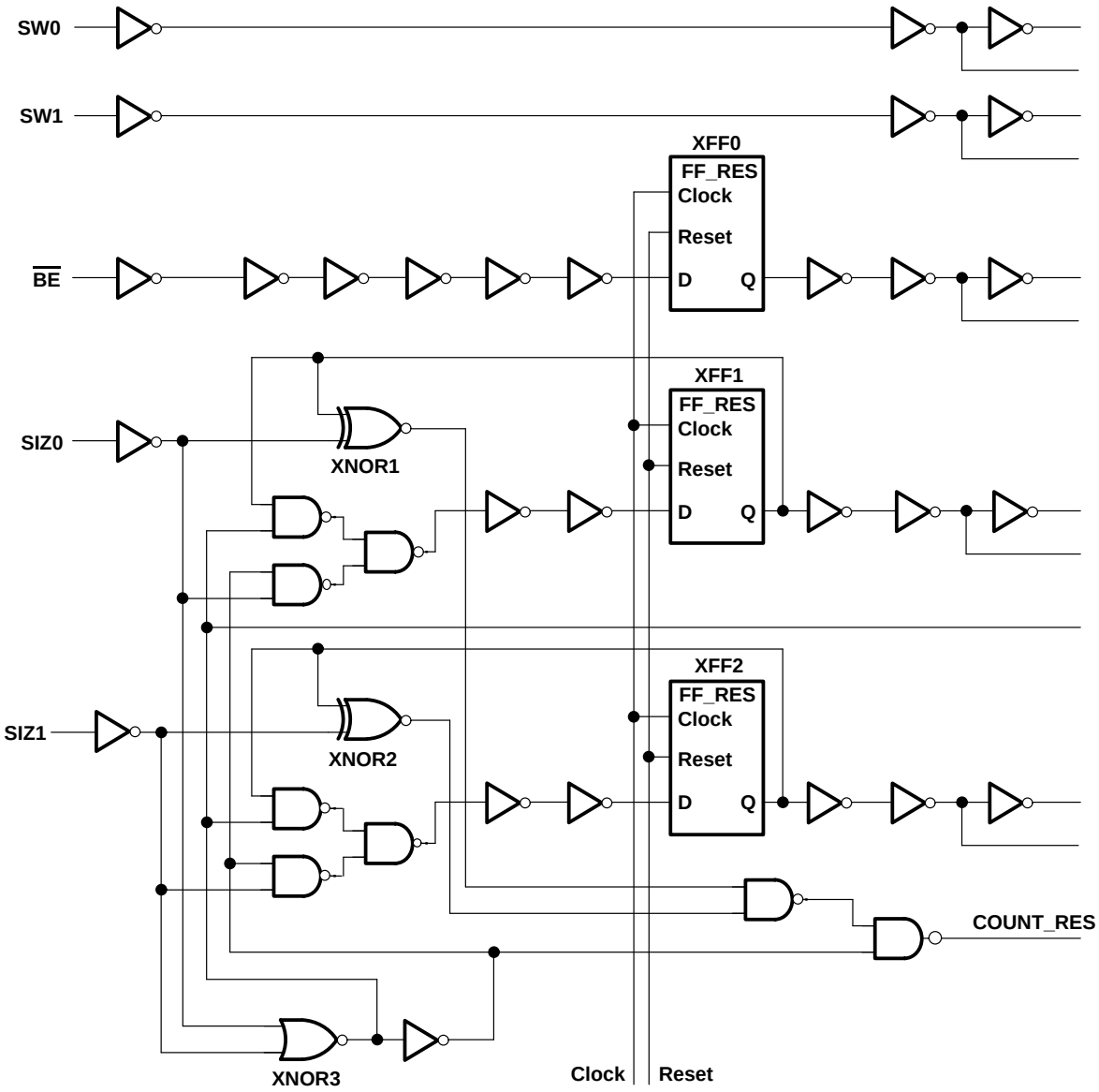


Figure 9. Size-Control Block

In the special case of mailbox operations, which use the size inputs to select mailbox mode on port B, it is essential not to interpret this as a change in bus size. When SIZ1 and SIZ0 inputs are driven high (mailbox mode), COUNT_RES is disabled (along with the counters) and the size registers are not updated (see XNOR2 in Figure 9). This allows mail operations to take place without disturbing any data transfers to/from the FIFO.

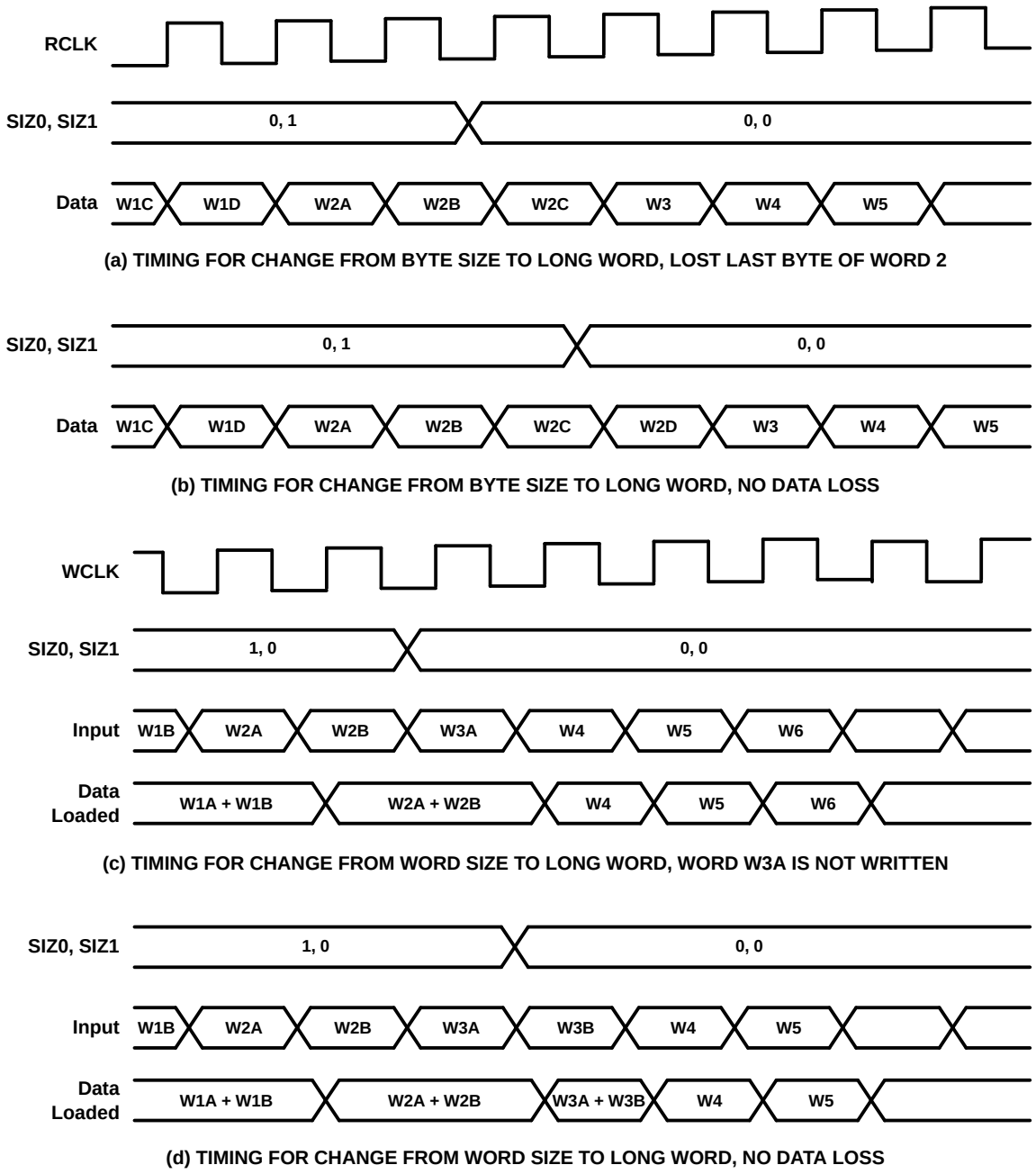


Figure 10. Dynamically Changing Byte-/Word-Size Transfers to Long-Word Data Transfers

Byte Swapping

Four different modes of byte-order arrangement (byte swap, word swap, byte-word swap, and no swap) can be performed with any port-B size selection. When byte swap is performed, the order of the bytes are rearranged within the long word but the bit order remains the same. The port-B swap-select (SW1 and SW0) inputs are used to achieve these byte-order arrangements. Table 2 lists the levels on the SW1 and SW0 input terminals and the respective size implemented. Figure 11 shows the different schemes of byte swap.

Table 2. Control of Byte-Swap Operation on Port B Using SW1 and SW0

SW1	SW0	BUS CONFIGURATION
L	L	No swap
L	H	Byte swap
H	L	Word swap
H	H	Byte-word Swap

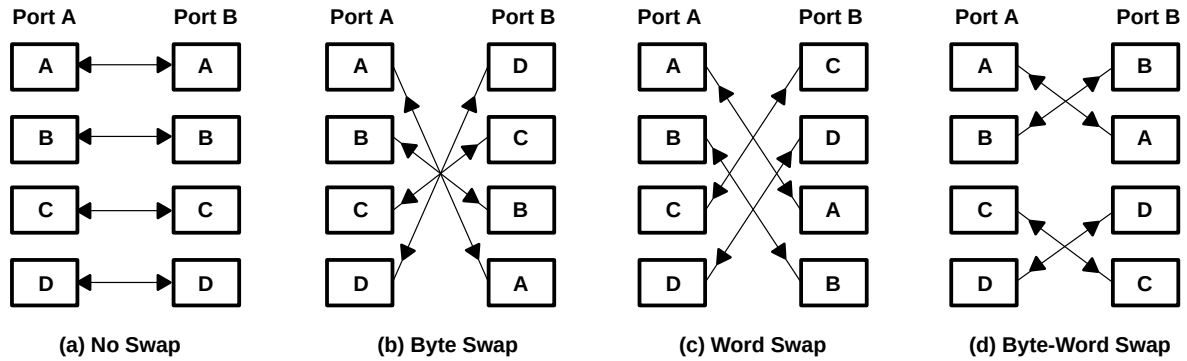


Figure 11. Byte-Swap for Long-Word-Size Data Transfers

The byte swap is performed with any port-B size selection. Byte-order arrangement is implemented by the levels on the port-B swap select inputs on a CLKB rising edge that reads a new long word from FIFO1 or writes a new long word to FIFO2. When long-word-size (36-bit) transfers are selected on port B, the byte-order arrangement can be changed on each clock cycle. On the other hand, when byte (9-bit) or word-size (18-bit) transfers are selected on port B, the byte order chosen on the first byte or word of a new long-word read from FIFO1 or written to FIFO2 is maintained until the entire long word is transferred, regardless of the swap-select input states during subsequent reads or writes. This implies that for byte- or word-sized data transfers, the byte-order arrangement can be changed only on the first byte or word data transfer of a new long-word read. Simultaneously performing a byte swap and bus size on port B results in the sequence of events shown in Figure 12.

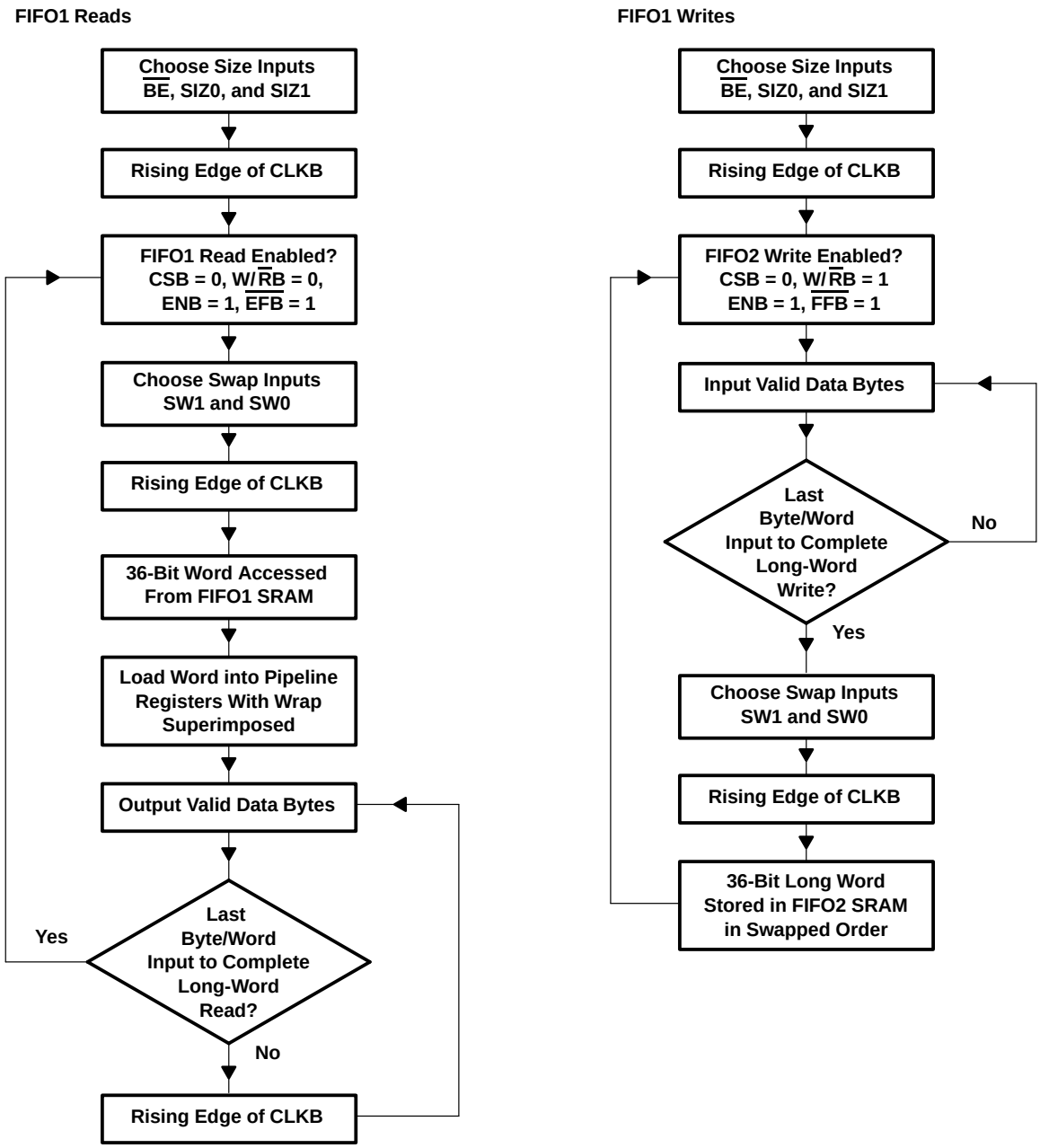


Figure 12. FIFO1 Data-Read and FIFO2 Data-Write Sequence During Simultaneous Bus-Sizing and Byte-Swapping Operations

Parity Generation and Checking

Parity Checking

The odd or even parity-checking function can be selected on both port A and port B using the ODD/$\overline{\text{EVEN}}$ input. Four parity trees examine the parity of incoming or outgoing data bytes on each port as shown in Figure 13. Port-A bytes are arranged as A0–A8, A9–A17, A18–A26, and A27–A35, with the most significant bit used as the parity bit. A parity error on each of these four bytes is indicated internally by a low signal on the individual-error ($\overline{\text{ER}}$) flag. These four $\overline{\text{ER}}$ outputs are combined to output a single parity-error ($\overline{\text{PEFA}}$) flag that indicates an error on one or more bytes on port A.

Similarly, port-B bytes are arranged as B0–B8, B9–B17, B18–B26, and B27–B35, with the most significant bit used as the parity bit. Again, a single parity-error ($\overline{\text{PEFB}}$) flag indicates an error on one or more bytes on port B. During the port-B sizing operation, the internal flag output for invalid data bytes is disabled; the parity-error flag indicates an error only on the bytes that are valid for the particular size selected. Parity checking on both ports is a passive operation; the port parity-error flags can be ignored if this feature is not desired.

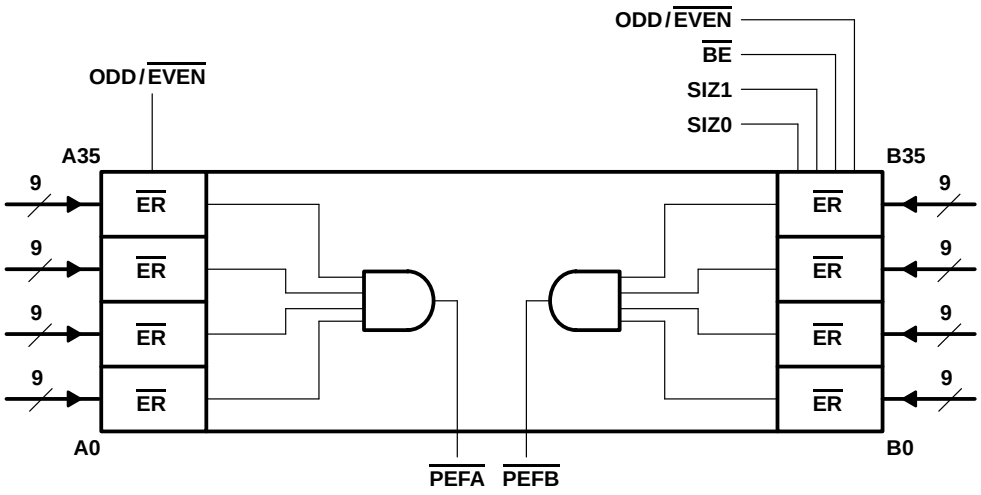


Figure 13. Parity-Checking Block Diagram

Parity Generation

Parity generation for port reads from the FIFO or mailbox can be selected using the parity-generate select-input terminal for that port (PGA or PGB). ODD/$\overline{\text{EVEN}}$ selects the type of parity generated. Port-A bytes are arranged as A0–A8, A9–A17, A18–A26, and A27–A35, with the most significant bit used as the parity bit. Port-B bytes are arranged as B0–B8, B9–B17, B18–B26, and B27–B35, with the most significant bit used as the parity bit. A write to the FIFO or mailbox stores all 36 bits regardless of the state of the parity generate select (PGA) input. When data is read from the FIFO, the lower eight bits of each byte are used to generate the parity bit as shown in Figure 14. If parity generation is selected (PGA = 1), the levels originally written to the most significant bits of each byte are substituted by the generated parity bit as the word is read to the outputs. Otherwise, the levels originally written to most significant bit of each byte are output.

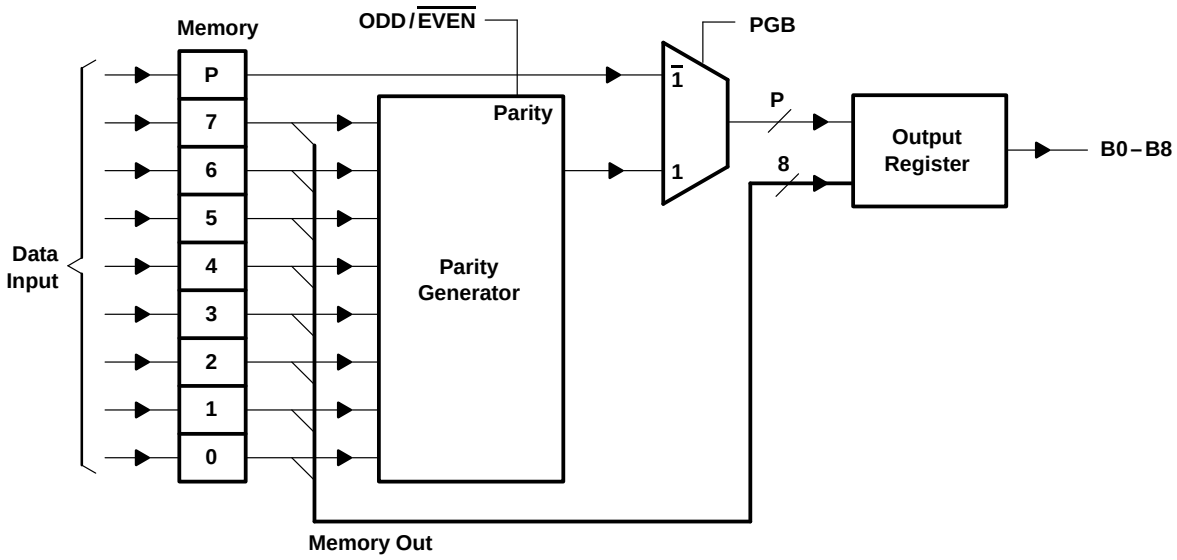


Figure 14. Parity-Generation Block Diagram

Parity generation on mailbox reads for a port is performed by the four parity trees used to check the parity on the inputs to the port. When odd- or even-parity generation is selected on a port-A or port-B read from the mailbox, the port parity-error ($\overline{\text{PEFA}}$ or $\overline{\text{PEFB}}$) flag is held high regardless of the state of the inputs A0–A35 or B0–B35. The generated parity does not change the contents of the mailbox register.

Internetworking

Internetworking is the process of connecting a variety of local-area-network (LAN) and wide-area-network (WAN) computer systems and related devices to build the communications infrastructure required to satisfy the needs of an organization. Bridges and routers are two key devices that provide internetworking capability (see Figure 15).

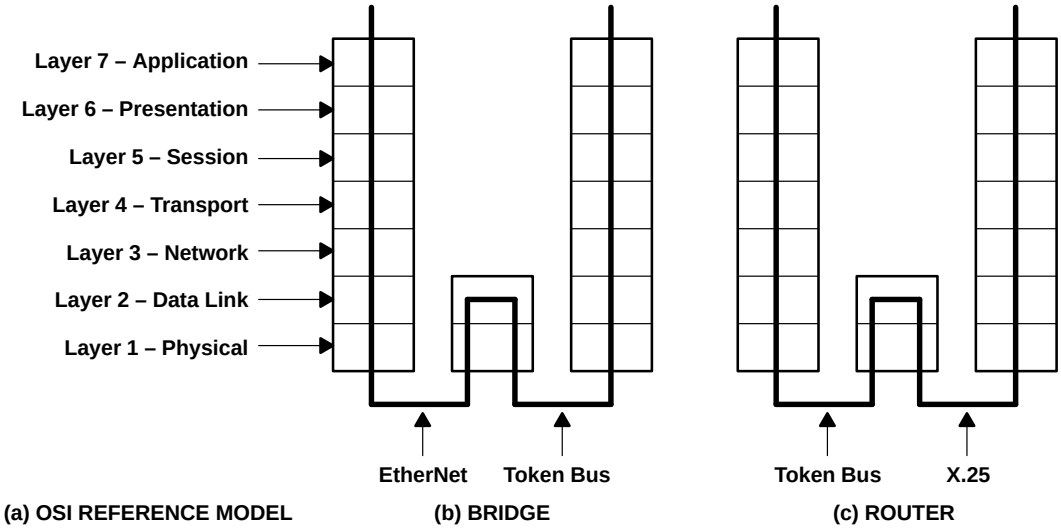


Figure 15. Bridge and Router Devices

Bridge and router designs are critical in meeting the increasing bandwidth requirements and volume of data transfer in existing network elements. The use of FIFOs in these designs provides a viable means for improving the system performance. The FIFOs

provide a link between the communications processors and buses operating at different speeds and data widths. The handshaking signals for control can provide operating speeds that match the bus-communication speeds.

A bridge operates at the data-link layer (layer 2) in the OSI model to connect two similar LANs. A bridge reads the destination address contained on each incoming packet and uses the address to transmit the packets or to ignore them. This process is known as address filtering. An example of a bridge design using multiple FIFOs is shown in Figure 16.

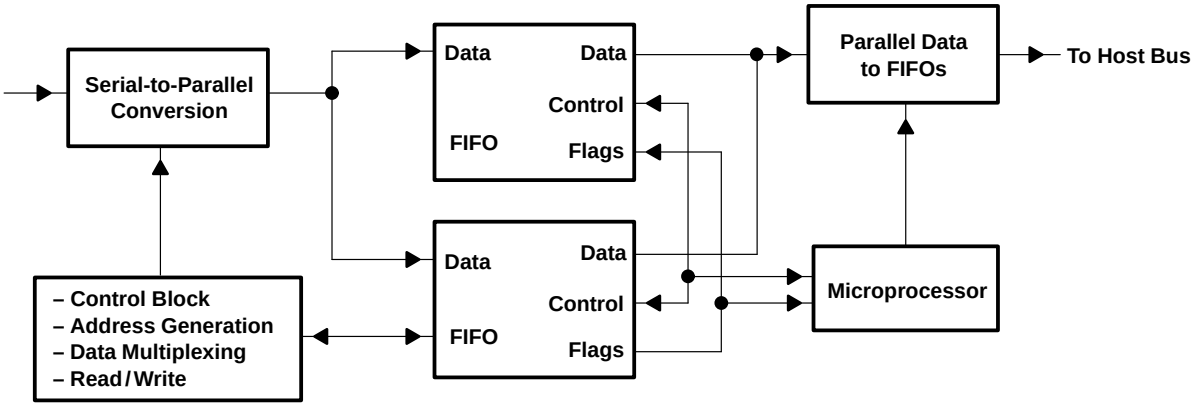


Figure 16. Implementation of a Bridge Using FIFOs

A router, on the other hand, operates at the network layer (layer 3) in the OSI model and can be used to connect two different networks; therefore, a router has the added complexity to perform frame conversion in addition to address filtering.

For example, in the situation where devices on a WAN communicate with the devices on a token-ring LAN, the interconnecting device is required to convert HDLC frames to token-ring frames and vice versa as shown in Figure 17.

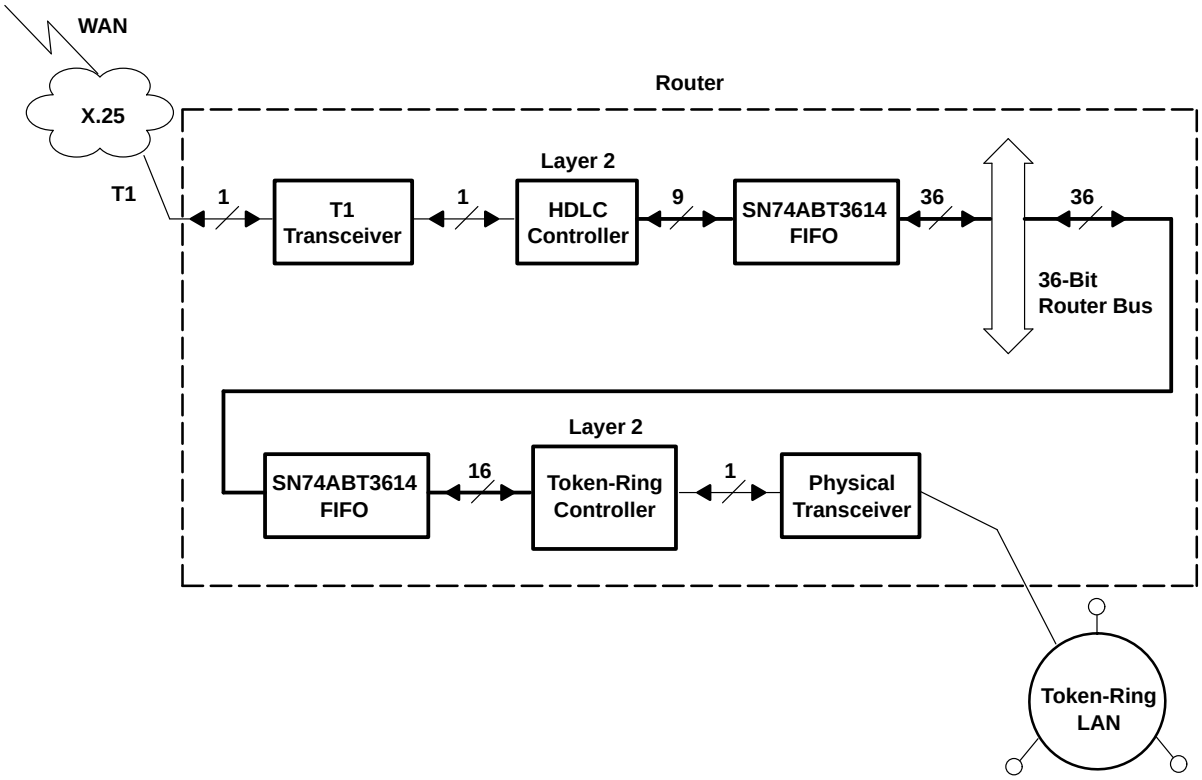


Figure 17. WAN to Token-Ring Router Using SN74ABT3614 FIFO

The layer-2 device provides the functions of flag generation and detection, zero-bit insertion and deletion, cyclic-redundancy-check (CRC) generation and detection, and abort generation and detection. The layer-2 device interfaces with the T1 transceiver on the line side. The SN74ABT3614 FIFO provides a bidirectional buffer between the layer-2 device and the high-speed router bus. In addition, if the layer-2 device has a 36-bit interface on the bus side, the bus-matching function on the SN74ABT3614 device is used to interface to a high-speed 36-bit bus. Similarly, the layer-2 device provides the necessary token-ring LAN protocols and links the token-ring LAN controller to the router bus. The SN74ABT3614 is used to match the data width and rates between the token-ring controller and the 36-bit router bus. A FIFO solution implemented in the hardware offers a speed advantage over the latency for a software setup performing the same functions.

Conclusion

The SN74ABT3614 is a clocked, bidirectional 64×36-bit member of TI internetworking family of FIFOs. This FIFO integrates the glue logic necessary to simplify design of communications networks. In addition to providing the traditional FIFO function of removing input/output bottlenecks, this FIFO provides full functionality for bus-sizing, byte-swapping, and parity-generation checking logic, and mailbox operations. These functions are necessary for effective communication between microprocessors, communications processors, and buses. The SN74ABT3614 FIFO can operate at frequencies up to 66 MHz and can provide data access times as fast as 11ns. The FIFO also is available in speed sorts to provide 33-MHz and 50-MHz operation.