Designing with CY7C436xx Synchronous FIFOs

Cypress Semiconductor offers four families of x36 FIFOs in 3.3V and 5V versions: The unidirectional with Bus Matching CY7C436x3AV/CY7C436x3, the bidirectional CY7C436x2AV /CY7C436x2, the bidirectional with Bus Matching CY7C436x4AV/CY7C436x4, and the Tri Bus with Bus Matching CY7C436x6AV/CY7C436x6. The Cypress web site at www.cypress.com provides a complete list of all configurations. All x36 families are available in a 128-pin TQFP package except for the bidirectional CY7C436x2AV/CY7C436x2, which comes in a 120-pin TQFP package.

The following is an outline of what is covered in this document:

- Modes of Operation: Standard vs. FWFT
- Reset: Master vs. Partial
- Bus Matching feature: Big Endian, size
- Programming: Serial, Parallel
- Mail Boxes
- Width and Depth Expansion
- Applications

Introduction

The x36 FIFOs are synchronous devices, which means all read and write operations are synchronized to the rising edge of a clock. Status flags are used to indicate the space availability on the RAM array. In Standard Mode, the Empty Flag ($\overline{EF}$), which is synchronized to the Read Clock (RCLK), gets asserted when the memory array is empty. This means that all data in the FIFO has been read out and no new data is available. The Full Flag ($\overline{FF}$), which is synchronized to the write clock (WCLK), gets asserted when the memory array is full. This means that no more data can be written into the memory.

In First Word Fall Through Mode, the Input Ready flag (IR) behaves similar to $\overline{FF}$. When IR is asserted, it indicates that a space in memory is available for data to be written. No writing is allowed when IR is inactive. The Output Ready (OR) flag is equivalent to the $\overline{EF}$. When OR is asserted, it indicates that data is available to be read out. The first word written to the FIFO in FWFT appears at the output without the requirement of a read operation. After that, it behaves as a Standard read operation.

Except for the unidirectional family, the arrangement of two independent dual-port SRAM FIFOs per package provides a great advantage in board space savings when these FIFOs are used to perform the functionality of two independent FIFOs such as the 1-Meg CY7C4285. It also provides the ability to interface two processors of different width without the use of external logic by using the Bus Matching feature. Today microprocessor-based systems come in a wide variety of bus widths: 9-, 16-, 32-, and even 64-bit. Communication between buses of different width is known as bus-matching. The x36 FIFOs are ideal devices for implementing the interface between x36 processors and wider (by expansion) or narrower processors using bus matching.

The application note “Understanding Sync FIFOs” explains the functionality and features of such devices and how the FIFO operations are synchronized to the clock transition. This application note mainly covers the features that may not exist in any other Cypress FIFO such as bus-matching, FWFT mode, and partial reset.

CY Standard vs. FWFT Mode

The difference between the Cypress Standard Mode (CY) and the First Word Fall Through (FWFT) Mode is in the first word written and read out of the FIFO. In CY Standard mode, the operation is similar to any sync FIFO operation. After writing the first word, in order to read it out with a free-running read clock, the second cycle will update the Empty Flag ($\overline{EF}$) status and the word appears at the output at the rising edge of the second cycle assuming Read Enable and Chip Enable are asserted. This is what is called a Read Operation (RO).

In FWFT mode, the first word written to an empty FIFO, which is indicated by the status of the IR flag, appears automatically on the output without the requirement of a Read Operation. This means less latency for reading data out of the FIFO. In order to access subsequent words, a RO is required. The status of the OR flag indicates the presence of data to be read.

Reading from the FIFO in FWFT Mode

For the 5V devices, the OR flag should be included in the logic reading from the FIFO due to some latency on this flag. When data is present in the FIFO, OR will be asserted, but immediately after the first word is read, OR will deassert for 3 read clock cycles before being asserted again until the FIFO is read to empty. For example, assuming 10 words have been written into the FIFO before the first read operation is performed. If read operation is performed in burst mode, i.e. all 10 words are read regardless of the state of OR flag, the data read will have 3 copies of the first word, while the final 3 words will still remain in the FIFO. In order to accurately access all 10 words with the FWFT mode, the FIFO should be read only when OR flag is asserted.

Note that this applies strictly to 5V devices only. Data can be accessed in burst mode for 3.3V devices for both CY standard and FWFT modes.

Master Reset vs. Partial Reset

Two kinds of resets are available on the x36 devices: Master and Partial Reset. As with any other Cypress FIFO, Master Reset brings the FIFO to its initial state: all read and write pointers back to location zero, output registers to all zeros, all programmed flags set back to default values and whatever programming was done will be lost. Each internal FIFO has its own Master Reset pin, MRS1 and MRS2.

Partial Reset will also initialize read and write pointers to the first location in memory, but it will retain all programming methods and programmable flag settings. In other words the FIFO memory gets emptied of any data without changing configuration settings. Similar to Master Reset, each internal FIFO has its own, independent Partial Reset pin: PRS1 and PRS2.

Bus Matching

Cypress x36 FIFOs come with the bus matching feature on Port B that would provide the ability to interface a x36 wide port to a x36, or x18, or x9 wide port without the need for any discrete bus arbitration or bus funneling control logic. Port B of these devices can be configured in a long-word (36-bits) format, word (18-bit) format, and byte (9-bit) format (for Tri Bus family CY7C436x6AV/CY7C436x6, Port B and Port C can be either in x18 or x9 format). The order of the bytes written in or read out in Port B depends on the BE (Big Endian) pin status. If BE is HIGH the format is said to be in the Big Endian configuration, else if BE is LOW then it would be in Little Endian format. The difference between the two formats depends on the placement of the most and least significant bytes in memory, as shown in *Figure 1*. Most Unix and internet protocols are Big Endian. Motorola 680x0 microprocessors, HP RISC, and Sun SuperSPARC processors are Big Endian too. The Intel 80x86 and Pentium and DEC Alpha RISC processors are Little Endian. *Table 1* shows how to configure Port B in any of these formats using BE, BM (Bus Match), and SIZE (Bus Size Select) signals. The level applied to these input signals is set during the LOW-to-HIGH transition of Master Reset ($\overline{MRS}$). Once a port is configured for a certain format, the setup has to be static throughout FIFO operation.

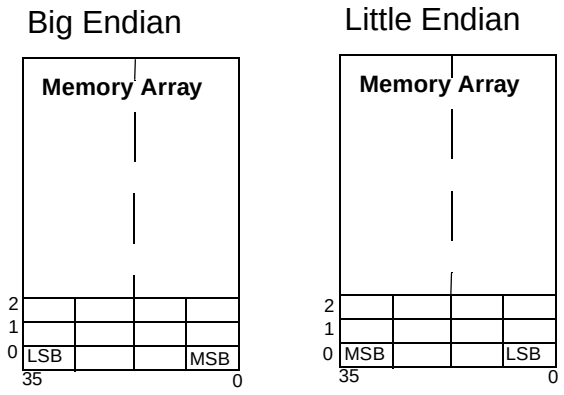

Figure 1. Big Endian vs. Little Endian Format

Table 1. Port B Bus Sizing

BM	BE	SIZE	Port-B format
L	X	X	x36
H	H	L	x18 Big Endian
H	L	L	x18 Little Endian
H	H	H	x9 Big Endian
H	L	H	x9 Little Endian

The following diagram shows the sequence in which the data is transferred from Port B to Port A and vice-versa, assuming the bidirectional with Bus Matching devices are being used.

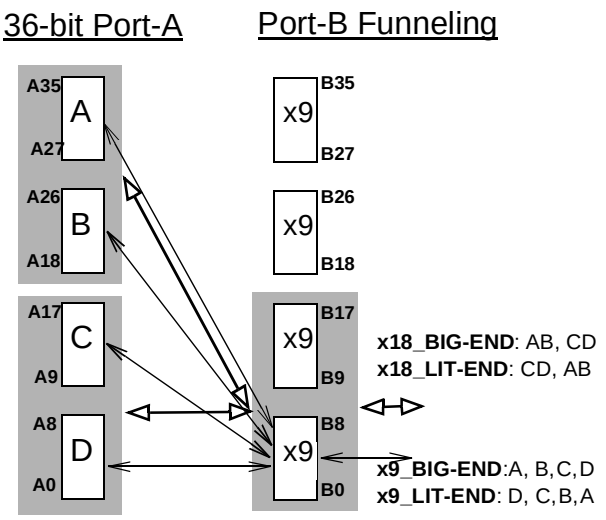

Figure 2. Port B Bus Funneling

When reading data in Byte or Word format, the unused outputs are LOW. If the interface is 36-bit to 36-bit then there will be no bus funneling and the data order at Port B will be equivalent to Port A.

Bus-matching operations are not available when transferring data via mailbox registers. Both the word (18-bit) and byte-size (9-bit) bus selections limit the width of the data bus that can be used for mail register operations. In this case, only those byte lanes belonging to the selected word- or byte-size bus can carry mailbox data. The remaining data outputs will be indeterminate. The remaining data inputs will be don't care inputs. For example, when a word-size bus is selected, then the mailbox data can be transmitted only between A0–A17 and B0–B17 data lines. When a byte-size bus is selected, then the mailbox data can be transmitted only between A0–A8 and B0–B8 data lines.

Width Expansion Configuration

As microprocessors move toward interfaces wider than 32 bits, Cypress x36 FIFOs can meet the demand by providing the ability to connect more than one of these FIFOs together (two x36 FIFOs would yield a 72-bit-wide bus). Status flags can be detected from any one FIFO with the exception of the $\overline{EF}$ and $\overline{FF}$ in the Cypress Standard mode and IR and OR in the FWFT mode. Due to the uncontrolled (variation) skew between Read Clock and Write Clock, it is possible that the deassertion/assertion of these flags can vary by one cycle between FIFOs. In Cypress Standard Mode (CY_MODE), this problem can be avoided by ANDing $\overline{EF}$ of every FIFO, and separately ANDing $\overline{FF}$ of every FIFO. In FWFT mode, composite flags can be created by ORing IR of every FIFO, and separately ORing OR of every FIFO.

Figure 3 shows a width expansion using two CY7C436xxAV/CY7C436xx (with the exception of Tri Bus family) devices. A0–A35 from each device form a 72-bit-wide input bus and B0–B35 from each device form a 72-bit-wide output bus. Additional FIFOs can be added to obtain larger width.

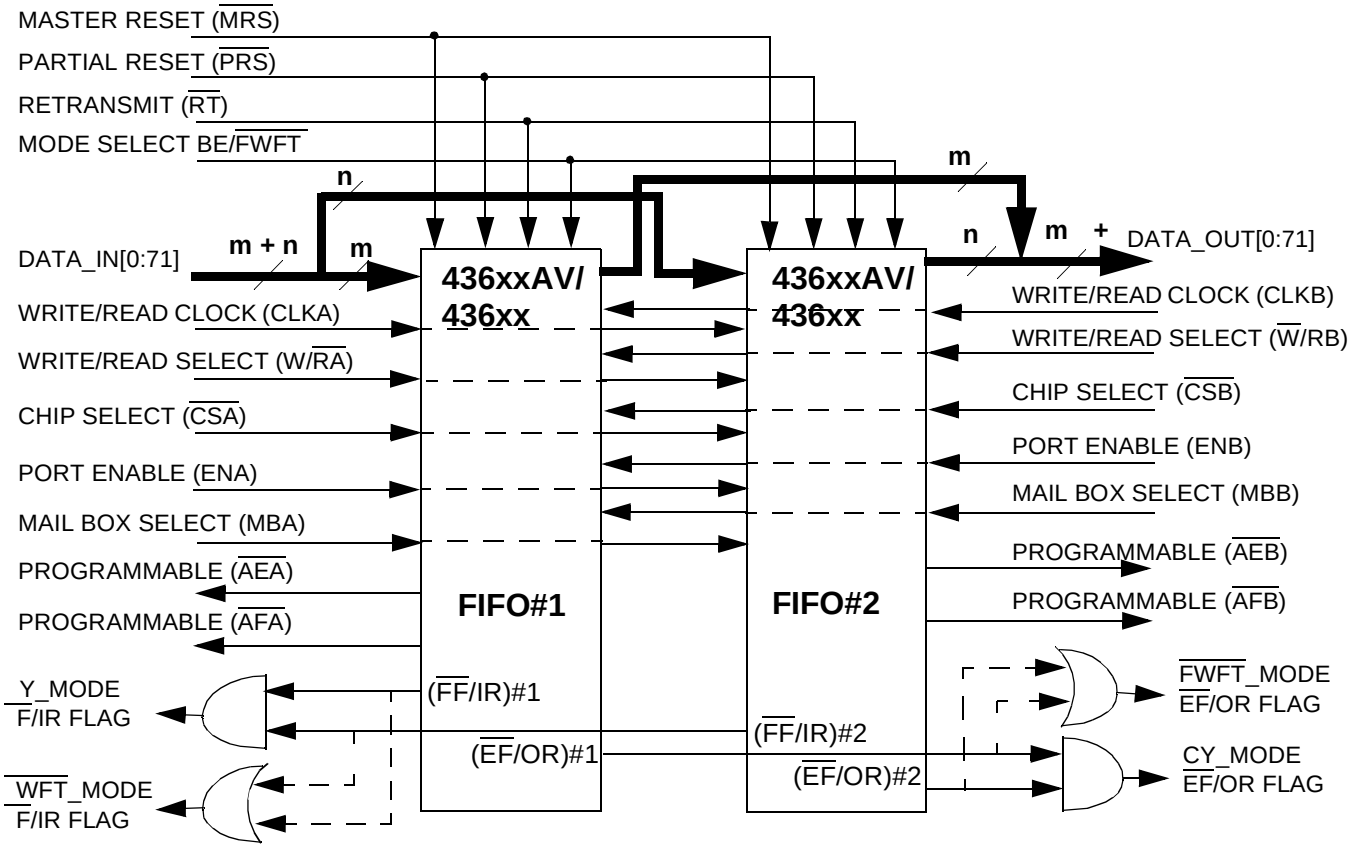

Figure 3. Block Diagram of Two CY7C436xxAV/CY7C436xx FIFOs in Width Expansion (72-bit Interface)

Depth Expansion Configuration (FWFT mode only)

The CY7C436xxAV/CY7C436xx can be easily configured to meet the demand for more depth in the memory array required by some applications. For example, two CY7C43682AV (1-Meg bidirectional FIFO) can be cascaded in depth to achieve a 2-Meg FIFO. In FWFT mode, the FIFOs can be connected in series (the data outputs of one FIFO connected to the data inputs of the next) with no external logic necessary.

Figure 4 shows a depth expansion using two CY7C436xxAV/CY7C436xx devices. Care should be taken to select FWFT mode during Master Reset for all FIFOs except the last one in the depth expansion configuration. The first word written to an empty configuration will pass from one FIFO to the next until it finally appears at the outputs of the last FIFO in the chain (no read operation is required for the first word during FWFT mode but the read clock of each FIFO must be free-running). Each time the data word appears at the outputs of one FIFO, that device's OR flag goes HIGH, enabling the write to the next FIFO in line. The first free location created by reading from a full depth expansion configuration will bubble up from the last FIFO to the previous one until it finally moves into the first FIFO of the chain. Each time a free location is created in one FIFO of the chain, that device's IR flag goes HIGH, enabling the preceding FIFO to write a word to fill it. The Reference Clock line should be tied to either CLKA or CLKB, whichever is faster. This configuration will result in data moving, as quickly as possible, to the end of the chain and free locations to the beginning of the chain.

Parallel and Serial Programming of Flags

Each of the four families of x36 FIFOs has a set of Almost Empty and Almost Full flags. Each flag is associated with its own offset register that can be programmed using two methods: parallel or serial—except for the bidirectional CY7C436x2AV/CY7C436x2 devices which only use the parallel programming method. For simplicity, the CY7C436x4AV family will be discussed; the reader should go back to the data sheets for the other parts.

The CY7C436x4AV has four flags, each with its respective offset register, as shown in *Table 2*. Three signals are available to perform the programming methods:

- FS1/SEN — Flag Offset Select 1 / Serial Enable
- FS0/SD — Flag Offset Select 2 / Serial Data
- SPM — Serial Programming Mode Select Pin

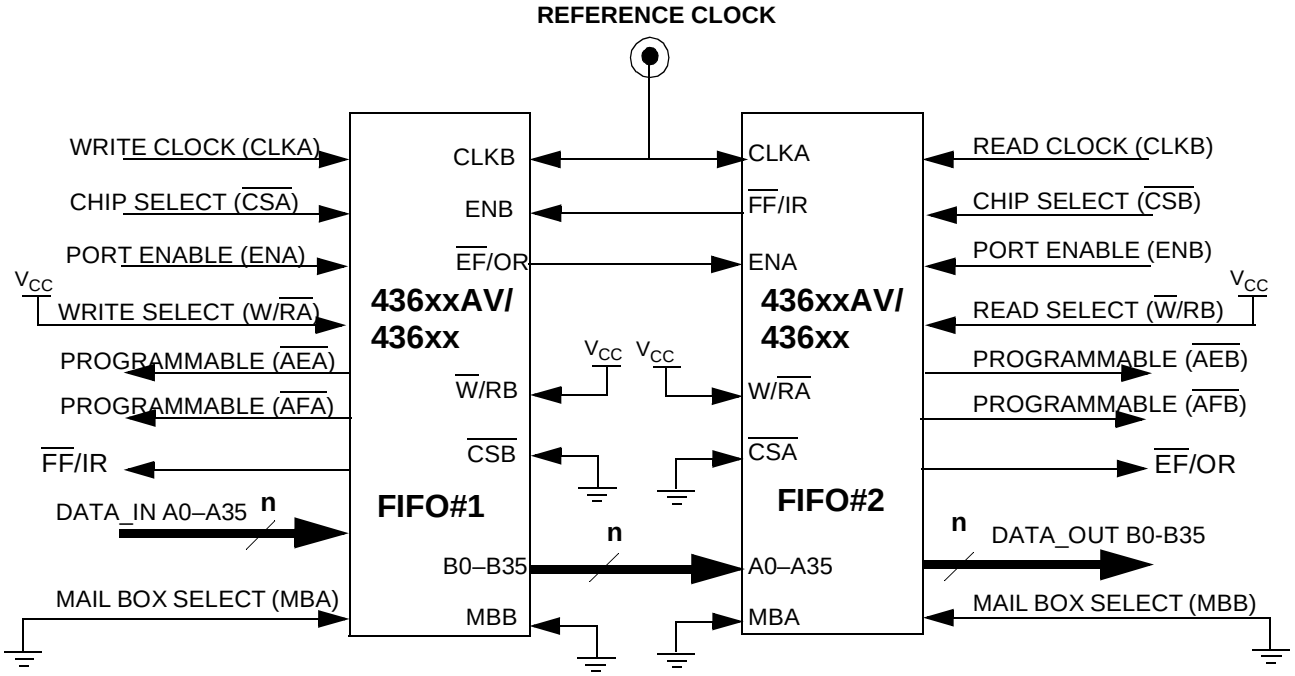

Figure 4. Block Diagram of Two CY7C436xxAV/CY7C436xx FIFOs in Depth Expansion

Table 2. Flags Offset Registers

Flag Name	Description	Offset Reg.
AEB	Port B Almost-Empty	X1
AFA	Port A Almost-Full	Y1
AEA	Port A Almost-Empty	X2
AFB	Port B Almost-Full	Y2

For parallel programming of the offset registers through Port A, a Master Reset is performed simultaneously on both internal FIFOs (MRS1 and MRS2). Set SPM HIGH, and FS0 and FS1 LOW during the LOW-to-HIGH transition of MRS1 and MRS2. After this reset is complete, the first four writes to FIFO1 (Port A side) do not store data in RAM but load the offset registers in the order Y1, X1, Y2, X2. The Port A data inputs used by the offset registers are:

- A0–A9 for CY7C43644V with valid values from 0 to 1023
- A0–A11 for Cy7C43664V with valid values from 0 to 4095
- A0–A13 for CY7C43684V with valid values from 0 to 16383

with the highest numbered input as the most significant bit of the binary number in each case.

For serial programming, master reset is initiated with SPM LOW, FS0/SD LOW, and FS1/SEN HIGH during the LOW-to-HIGH transition of MRS1 and MRS2. Then the register values are loaded bit-wise through the SD input on each LOW-to-HIGH transition of CLKA with the SEN input LOW. The number of bit-writes needed to complete the programming of the registers in the order Y1, X1, Y2, X2 are:

- 40 for CY7C43644V with valid values from 0 to 1023
- 48 for CY7C43664V with valid values from 0 to 4095
- 56 for CY7C43684V with valid values from 0 to 16383

The first bit-write stores the most significant bit of the Y1 register and the last bit-write stores the least significant bit of X2 register.

If no programming method is selected, the offset registers can automatically be loaded by one of three preset (default) values of 8, 16, and 64. The data sheet has a table that lists the status of the control pins when selecting any of these values.

Programmable Flags Uncertainties

If read and write operations are performed on the FIFO simultaneously at the programmable flags' boundary, the programmable flag may exhibit some degree of uncertainties. There are slight differences between the 5V and 3.3V devices' flag uncertainties. 5V devices have up to –2 words of uncertainty on the assertion edge (H to L), and –1 to +2 words of uncertainties when the flags deassert (L to H). 3.3V devices' assertion edge is always accurate to the programmed/default value, but have up to +2 words of uncertainty on the deassertion edge.

For instance, assume that the almost empty (AE) and almost full (AF) flags of a 5V device are programmed to assert at 8 words. If equal amount of writes and reads were to occur when the FIFO content is at 8 words and when the FIFO is at 8 words from full, AE may assert when there are 6 to 8 words left in the FIFO and may deassert when the number of words in the FIFO is anywhere between 7 and 10. AF may deassert when there are 7 to 10 spaces left in the FIFO and assert when there are 6 to 8 spaces left in the FIFO.

For the 3.3V device, assume again that the Almost Empty ($\overline{AE}$) and Almost Full ($\overline{AF}$) flags are programmed to assert at 8 words, then $\overline{AE}$ will always assert when its content has exactly 8 words, but the $\overline{AE}$ flag may deassert when the FIFO content is at 9 or 10 words. For the $\overline{AF}$ flag, deassertion may occur when the FIFO has 9 to 10 spaces left while assertion will always occur exactly when there are 8 spaces left in the FIFO.

The following table shows the flag uncertainties for 5V and 3.3V devices when programmed to 8 words ("N" indicates the full depth of the FIFO).

Table 3. Comparison of Flag Uncertainties (all programmed to assert at 8)

436xx (5V)	436xxAV (3.3V)	AE	AF
6 to 8	8	**H->L**	H
7 to 10	9 to 10	**L->H**	H
N-10 to N-7	N-10 to N-9	H	**L->H**
N-8 to N-6	N-8	H	**H->L**

Recommended Design Practice

Depending on individual customer design, the uncertainties of the flags may or may not be a concern. If precision is not required of the flags then it is not an issue, otherwise, care should be taken to ensure data integrity. Note that under normal operating conditions where simultaneous reads and writes do not occur at the flag boundary, such uncertainties will not happen.

5V $\overline{AF}$ Assertion Edge

This is the situation where a certain amount of data is already present in the FIFO, and more data is being *written* into it, thus crossing the almost full boundary from not-almost-full to almost-full state.

If the application requires the FIFO to guarantee Y number of spaces when $\overline{AF}$ asserts, i.e. Y number of words are intended to be written into the FIFO when $\overline{AF}$ asserts, then $\overline{AF}$ should be programmed to Y+2 spaces to guarantee the availability of Y spaces in the FIFO. For example, if 20 words are intended to be written into the FIFO upon the assertion of $\overline{AF}$, then $\overline{AF}$ should be programmed to 22 (uncertainty factor is –2, thus $\overline{AF}$ may assert anywhere between 20 to 22 spaces). This way when $\overline{AF}$ does assert, the FIFO is guaranteed to have at least 20 empty spaces.

If the application requires the FIFO to guarantee that its content has at least [depth-Y] words when $\overline{AF}$ asserts, then $\overline{AF}$ should be programmed to Y to have such guarantee. For example, assuming the FIFO has a depth of 4096 words, if $\overline{AF}$ is required to assert when FIFO content achieves 3968 words, $\overline{AF}$ should be programmed to 128 (uncertainty factor is –2, thus $\overline{AF}$ may assert anywhere between 126 to 128 spaces). Thus when $\overline{AF}$ asserts, its content would have at least 3968 words. Note that a couple of additional words may need to be written into the FIFO at 3968 to ensure $\overline{AF}$ asserts.

5V $\overline{AF}$ Deassertion Edge

This is the situation where the FIFO content is very close to full, and is being *read* such that the FIFO content is decreasing and is crossing the almost full boundary from almost-full state to not-almost-full state.

If the application requires the FIFO to guarantee the availability of Y spaces upon the deassertion of $\overline{AF}$, and reads either continuously or random number of words from the FIFO each time, then $\overline{AF}$ should be programmed to Y+1. For instance, if the FIFO must have 20 spaces available when $\overline{AF}$ deasserts, then $\overline{AF}$ should be programmed to 21 (uncertainty factor is –1 to +2, thus $\overline{AF}$ may deassert anywhere between 20 to 23 spaces). Additional reads may be required at 21-space boundary for $\overline{AF}$ to deassert.

However, if the FIFO is read and written in the form of packets (where a packet is a user-defined, fixed number of words), and $\overline{AF}$ deassertion is used as a trigger to indicate a certain number of packets have been read, then $\overline{AF}$ should be programmed to Y–2 to guarantee the $\overline{AF}$ rising edge. For instance, if a packet is defined to be 128 words, and $\overline{AF}$ is required to deassert when 2 packets (i.e., 256 words) have been read from a full FIFO, then $\overline{AF}$ should be programmed to 254 words (uncertainty factor is –1 to +2, thus $\overline{AF}$ may deassert anywhere between 253 and 256). This way $\overline{AF}$ is guaranteed to deassert when these 2 packets have been read.

If the application requires the FIFO to guarantee the availability of [depth–Y] words upon the deassertion of $\overline{AF}$, then $\overline{AF}$ should be programmed to Y–2 as well.

5V $\overline{AE}$ Assertion Edge

This is the situation where most of the FIFO content is empty, and the FIFO is being *read*, such that its content is crossing the almost empty boundary from not-almost-empty state to almost-empty state.

If the application requires the FIFO to guarantee to have a minimum of X number of words when $\overline{AE}$ asserts, then $\overline{AE}$ should be programmed to X+2 words. For instance, if 16 words are expected to be read from the FIFO upon assertion of $\overline{AE}$ flag, then $\overline{AE}$ should be programmed to 18 words (uncertainty factor is –2, thus $\overline{AE}$ may assert anywhere between 16 and 18 words). Thus when $\overline{AE}$ asserts, the FIFO is guaranteed to have a minimum of 16 words.

If the application requires the FIFO to guarantee [depth–X] spaces when $\overline{AE}$ asserts, such that this amount of new data can be written into the FIFO, then $\overline{AE}$ should be programmed to X. For example, assuming the FIFO has a depth of 4096 words, if $\overline{AE}$ is required to assert when FIFO has at least 3072 empty spaces, $\overline{AE}$ should be programmed to 1024 (uncertainty factor is –2, thus $\overline{AE}$ may assert anywhere between 1022 to 1024 spaces). Thus when $\overline{AE}$ asserts, its would have at least 3072 spaces. Note that a couple of additional reads may be required at 3072 boundary to ensure $\overline{AE}$ asserts.

5V $\overline{AE}$ Deassertion Edge

This is the situation where the FIFO content is very close to empty, and is being *written* such that its content is increasing and is crossing the almost empty boundary from almost-empty state to not-almost-empty state.

If the application requires the FIFO to guarantee to have at least X number of words upon the deassertion of $\overline{AE}$, and writes either continuously or random number of words into the FIFO each time, then $\overline{AE}$ should be programmed to X+1. For instance, if the FIFO must have 20 words available when $\overline{AE}$ deasserts, then $\overline{AE}$ should be programmed to 21 (uncertainty factor is –1 to +2, thus $\overline{AE}$ may deassert anywhere between

20 to 23 spaces). Additional writes may be required at 21-word boundary for $\overline{AE}$ to deassert.

However, if the FIFO is read and written in the form of packets and $\overline{AE}$ deassertion is used as a trigger to indicate a certain number of packets have been written, then $\overline{AE}$ should be programmed to X–2. For instance, if a packet is defined to be 128 words, and $\overline{AE}$ is required to deassert when 2 packets (i.e., 256 words) have been written into an empty FIFO, then $\overline{AE}$ should be programmed to 254 words (uncertainty factor is –1 to +2, thus $\overline{AE}$ may deassert anywhere between 253 and 256). This way $\overline{AE}$ is guaranteed to deassert when these 2 packets have been written.

If the application requires the FIFO to guarantee the availability of [depth–X] spaces upon the deassertion of $\overline{AE}$, then $\overline{AE}$ should be programmed to X–2 as well.

3.3V $\overline{AF}$ Deassertion Edge

This is, again, the situation where the FIFO content is very close to full, and is being *read* such that the FIFO content is decreasing and is crossing the almost full boundary from almost-full- state to not-almost-full state.

If the application requires the FIFO to guarantee the availability of Y spaces upon the deassertion of $\overline{AF}$, and reads either continuously or random number of words from the FIFO each time, then $\overline{AF}$ should be programmed to Y. For instance, if the FIFO must have 20 spaces available when $\overline{AF}$ deasserts, then $\overline{AF}$ should be programmed to 20 (uncertainty factor is +2, thus $\overline{AF}$ may deassert anywhere between 20 to 22 spaces). Additional reads may be required at 20-space boundary for $\overline{AF}$ to deassert.

However, if the FIFO is read and written in the form of packets and $\overline{AF}$ deassertion is used as a trigger to indicate a certain number of packets have been read, then $\overline{AF}$ should be programmed to Y–2 to guarantee the $\overline{AF}$ rising edge. For instance, if a packet is defined to be 128 words, and $\overline{AF}$ is required to deassert when 2 packets (i.e., 256 words) have been read from a full FIFO, then $\overline{AF}$ should be programmed to 254 words (uncertainty factor is +2, thus $\overline{AF}$ may deassert anywhere between 254 and 256). This way $\overline{AF}$ is guaranteed to deassert when these 2 packets have been read.

If the application requires the FIFO to guarantee the availability of [depth–Y] words upon the deassertion of $\overline{AF}$, then $\overline{AF}$ should be programmed to Y–2 as well.

3.3V $\overline{AE}$ Deassertion Edge

This is, again, the situation where the FIFO content is very close to empty, and is being *written* such that its content is increasing and is crossing the almost empty boundary from almost-empty state to not-almost-empty state.

If the application requires the FIFO to guarantee to have at least X number of words upon the deassertion of $\overline{AE}$, and writes either continuously or random number of words into the FIFO each time, then $\overline{AE}$ should be programmed to X. For instance, if the FIFO must have 20 words available when $\overline{AE}$ deasserts, then $\overline{AE}$ should be programmed to 20 (uncertainty factor is +2, thus $\overline{AE}$ may deassert anywhere between 20 to 22 spaces). Additional writes may be required at 20-word boundary for $\overline{AE}$ to deassert.

However, if the FIFO is read and written in the form of packets and $\overline{AE}$ deassertion is used as a trigger to indicate a certain number of packets have been written, then $\overline{AE}$ should be programmed to X–2. For instance, if a packet is defined to be 128 words, and $\overline{AE}$ is required to deassert when 2 packets (i.e., 256 words) have been written into an empty FIFO, then $\overline{AE}$ should be programmed to 254 words (uncertainty factor is +2, thus $\overline{AE}$ may deassert anywhere between 254 and 256). This way $\overline{AE}$ is guaranteed to deassert when these 2 packets have been written.

If the application requires the FIFO to guarantee the availability of [depth–X] spaces upon the deassertion of $\overline{AE}$, then $\overline{AE}$ should be programmed to X–2 as well.

3.3V $\overline{AF}$ and $\overline{AE}$ Assertion Edge

Both of these edge will assert exactly at the programmed value.

Mailbox Registers

Mailbox registers are used to bypass the FIFO memory array in order to communicate a command or control information from one port to the other. There are two Mailbox registers: Mail-1 register associated with FIFO1 and Mail-2 register associated with FIFO2. Each Mailbox has a port select pin that is active HIGH. These are MBA, MBB, and MBC for Port A, Port B, and Port C respectively. MBC is only used for Port C of Tri Bus devices (CY7C436x6AV/CY7C436x6) which will choose the Mail-2 register for Port C write operation. The remaining discussion will only target the unidirectional, bidirectional, and bidirectional with Bus Matching devices.

The Mailbox Select (MBA, MBB) inputs choose between a mail register and a FIFO for a port data transfer operation. The usable width of both Mail-1 and Mail-2 registers matches the selected bus size for Port B. When writing data to a mail register, the unused data lines (if it is not 36-bit) will be don't care. When reading data from a mail register, the unused data lines will be indeterminate. *Table 4* shows the operation of the mailbox and the status of the flags. MBF1 is the Mail-1 Register Flag and is set LOW when Port A writes to it. It is cleared (HIGH) when Port B reads data from the Mail-1 register. MBF2 is the Mail-2 Register Flag and is set LOW when Port B writes to it. It is cleared (HIGH) when Port A reads data from Mail-2 register.

Table 4. MailBox Operations

CLKA	CLKB	MBA	MBB	MBF1	MBF2	Operation
L-H	L-H	L	L	H	H	Data passed through memory array
L-H	L-H	H	-	L	-	Port-A Write to Mail-1 Register
L-H	L-H	-	H	-	L	Port-B Write to Mail-2 Register
L-H	L-H	-	H	H	-	Port-B Read Mail-1. Clears $\overline{MBF1}$
L-H	L-H	H	-	-	H	Port-A Read Mail-2. Clears $\overline{MBF2}$

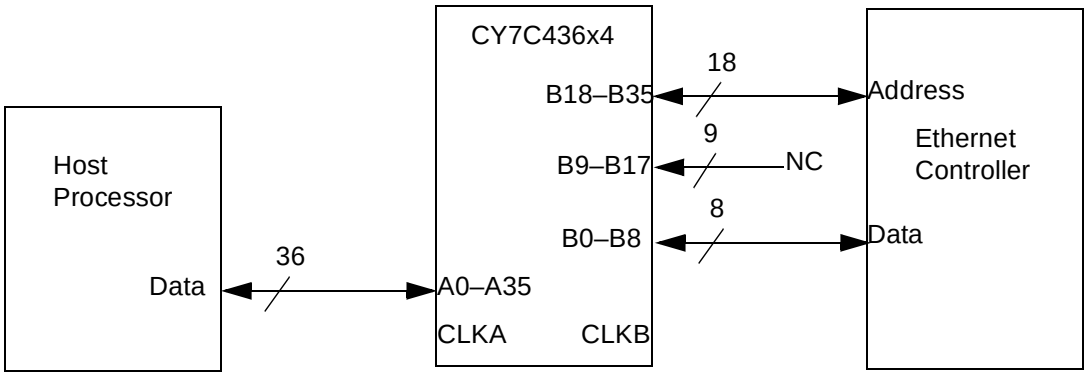

Figure 5. Broadband Communication Application Using CY7C436x4

When data outputs of a port are active, the data on the bus comes from the FIFO output register when the port Mailbox Select is LOW, and from the mail register when the port Mailbox Select input is HIGH. The data in a mail register remains intact after it is read and changes only when new data is written to the register. Also, attempted writes to a mail register are ignored while the mail flag (MBF1 or MBF2) is LOW.

Broadband Communications Application

Figure 5 shows an example of using the bidirectional with Bus Matching FIFOs to interface a 32-bit host processor to an 8-bit Ethernet microcontroller. The use of the CY7C436x4 provides a simple and cost-effective solution for a high-speed data communications application such as this. Port B of the 7C436x4 is configured in byte size 9-bit wide).

In the above application, the microcontroller provides information to the host processor by connecting its 16-bit address line to the B18–B35 data lines, and connecting the controller 8-bit data line to the B0–B8 data lines of the FIFO. The connection to the host processor is a direct connection through the A0–A35 data lines of Port A of the FIFO. Therefore, the address and data cycle consists of a 16-bit address followed by four 32-bit long-words. By configuring Port B to be in Little-Endian format (BE is LOW) and in Byte size (BM is HIGH and SIZE is HIGH), the 16-bit address must be read first by the processor during the first long-word read cycle as shown in *Figure 6*. *Figure 6* also shows the sequence of the long-word data read by the processor. It will take 17 clock cycles (17 CLKB transitions) to complete writing ADD0–ADD15 and Byte 0–Byte 16 to FIFO Port B by the Ethernet microcontroller, whereas it will take 5 cycles (5 CLKA transitions) to complete reading the information from FIFO Port A. The microcontroller first writes the 16-bit address to the most significant part of Port B of the FIFO with B26–B35 being the most significant Byte (Little-Endian). Then the 8-bit data is written to the least significant part of port-B (B0–B8) with Byte 0 written first then Byte 1 and so forth till Byte 15. Four 8-bit data bytes are combined to form a 32-bit long-word that the host processor will read from Port A of the FIFO in the sequence shown in *Figure 6*.

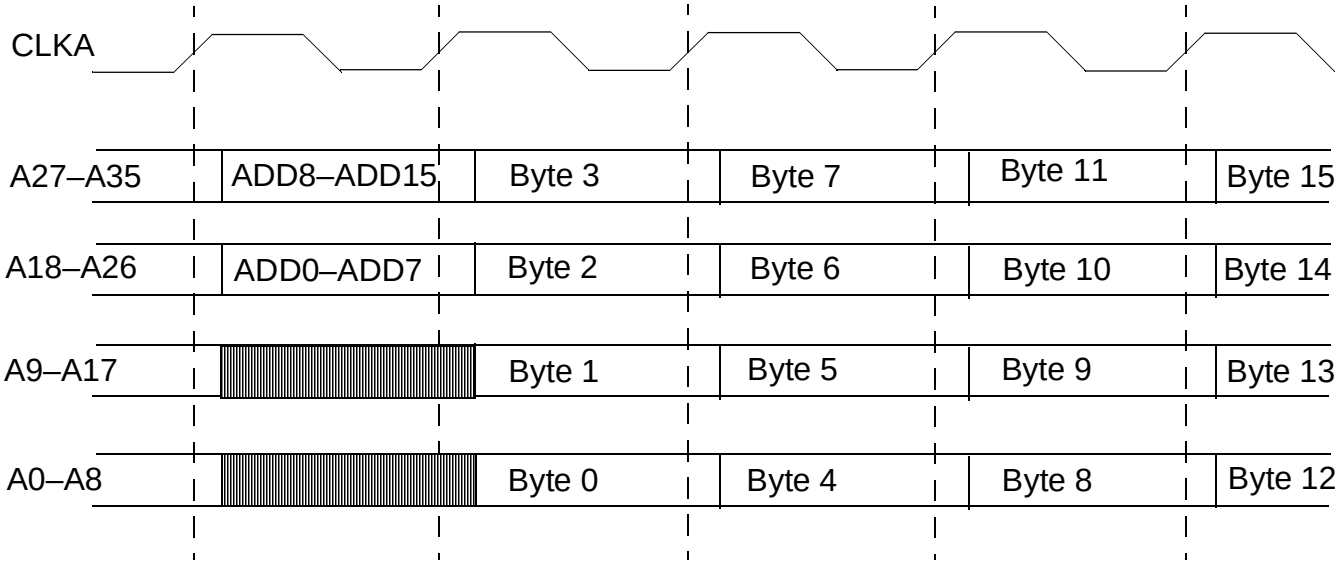

Figure 6. Read From FIFO Port A By Host Processor

HOTLink to PCI Interface

The PCI interface used in this application is a target only interface accomplished using the CY7C37256 256-macrocell CPLD. The target interface has standard PCI bus interface, as well as a local 32-bit FIFO interface. The interface to the FIFO is done using Port A of the Tri Bus x36 CY7C436x6 FIFO with Bus Matching. Port A of the CY7C436x6 FIFO is a bidirectional, 36-bit-wide bus. Port B of the FIFO is an 18-bit-wide bus with the bus-matching feature used to transfer data from the PCI bus to the HOTLink Transmitter (CY7B923). Port C of the FIFO is also an 18-bit-wide bus with the bus-matching feature used to transfer data from the HOTLink Receiver (CY7B933) to the PCI bus. Both Port B and Port C are configured in byte-size Little-Endian format by connecting the BE pin to GND and both SIZEB and SIZEC pins to V_{CC}. This PCI interface also has an interrupt generating capability. When the local interrupt is asserted for one PCI bus cycle, INTA on the PCI side is asserted. The interrupt is used to notify the PCI-bus master/processor that a new frame of data has been received and is available from the HOTLink interface card. *Figure 7* shows the major blocks needed to implement the PCI bus interface to the HOTLink Transmitter and Receiver. Such an interface will enable two PCs to communicate at 330 Mbps with 8B/10B encoding over either copper cables or optical cables.

The 32-bit data written into the x36 FIFO will be read by the HOTLink transmitter over four clock cycles (at the rising edges of CLKB) in byte size. The TX_Hotlink then will perform 8B/10B encoding and shift the data out serially. The data will be transmitted onto the physical medium either through a fiber-optic interface or a STP copper connector. When the host wishes to send a command byte, the host will write to a different address on the card designated for command passing. By writing to this address, the ninth bit (B8) accompanying this command byte will be set to indicate a command.

The receive path is implemented using the HOTLink Receiver (CY7B933) as the serial data recovery device. Serial data coming into the system from the physical medium will be converted into single-ended or differential PECL (Pseudo ECL) signal. The RX_Hotlink will perform clock to data recovery, byte alignment, deserialization, and 8B/10B decoding.

The 8-bit size received data is then written into the CY7C436x6 FIFO through Port C, which is configured in byte-size and Little-Endian format. Data stored in the FIFO is then read by the PCI interface implemented using the CY7C37256 PLD through Port A of the FIFO. The PCI bus reads data in 32-bit long-word format which means for every four clock cycles of CLKC (Port C clock) required to write data into the FIFO, it will only take one clock cycle of CLKA (Port A clock) to read the data out. *Figure 8* shows the sequence of the data going into the FIFO and coming out of the FIFO. t_{D1} is the skew time between CLKA and CLKB plus two CLKB clock cycles required to update the Empty Flag ($\overline{EFB}$), and t_{D2} is the skew time between CLKC and CLKA plus two CLKA clock cycles needed to update the Empty Flag ($\overline{EFA}$).

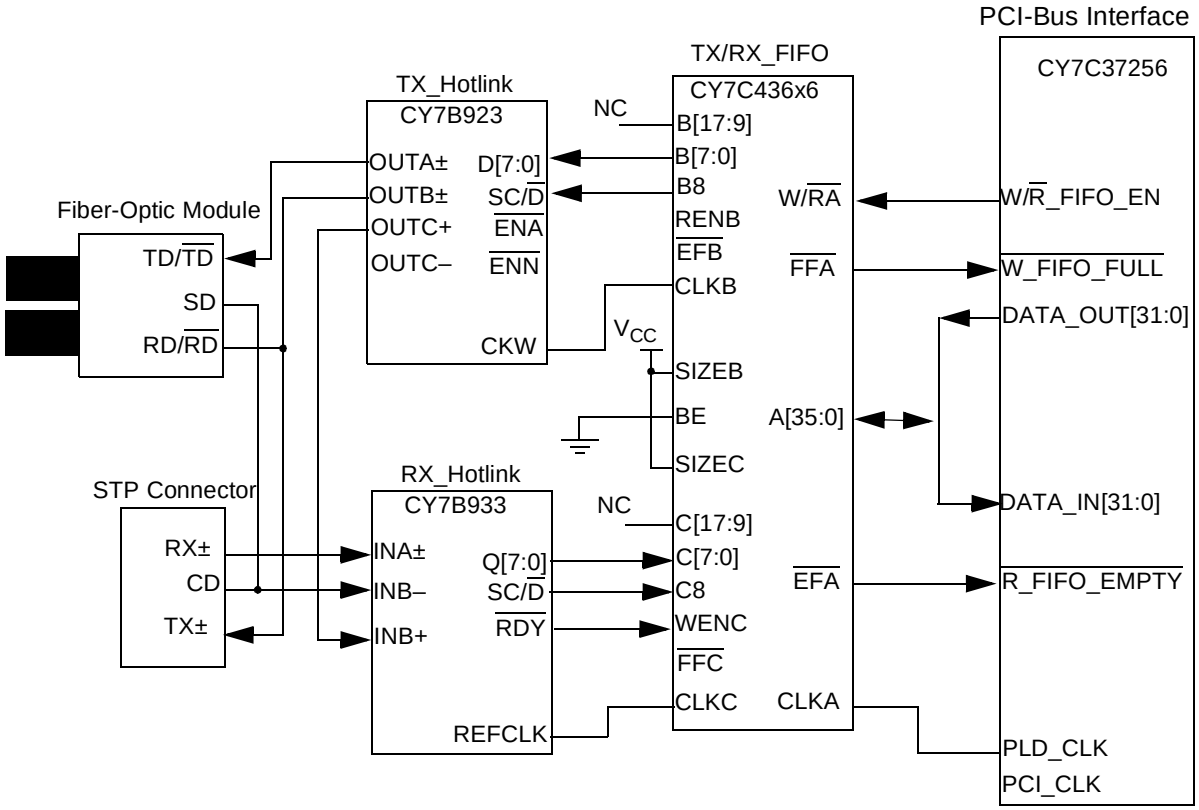

Figure 7. HOTLink to PCI Interface Using CY7C436x6V

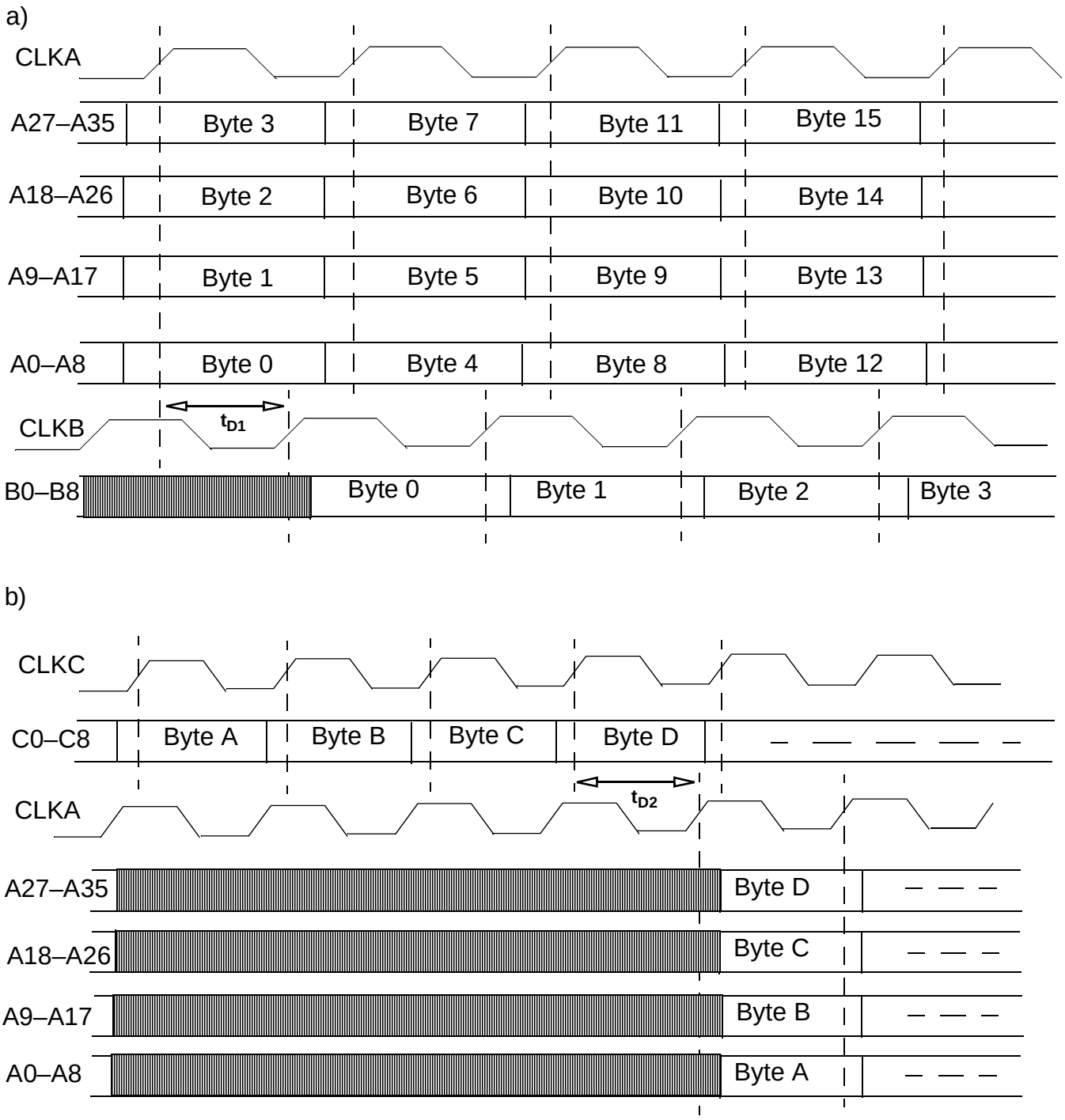

Figure 8. CY7C436x6V Data Flow; a) Port A to Port B, b) Port C to Port A

When a command word is read out from the FIFO (designated by the ninth bit of every byte since bit C8 is connected to the RX_Hotlink SC/D signal), the PCI controller will interrupt the host. The host will perform a normal read cycle, reading the command word and in turn clearing the interrupt. Normal data transfer resumes after the interrupt is cleared.

In this example, the use of the CY7C436x6 Tri Bus with Bus Matching FIFO replaced the what would have been eight 9-bit-wide FIFOs (four of them would have been expanded in width to serve as the transmit data buffer between the PCI bus and the HOTLink Transmitter, and the other four 9-bit-wide FIFOs would have been cascaded in width to perform as the receive data buffer between the HOTLink Receiver and the

PCI bus). Extra logic would have been required to control the outputs of the individual FIFOs and provide word alignment.

Conclusion

The CY7C436xxAV/CY7C436xx families of Cypress Semiconductor Synchronous FIFOs provide a great interface and effective communication capability between microprocessors, communications processors, and buses. These FIFOs integrate the glue logic necessary to simplify the design of communications networks. In addition to providing the traditional FIFO function of removing input/output bottlenecks, these FIFOs provide full functionality for bus-sizing, byte format, and mailbox operations. These FIFOs provide the system designer with a board saving solution and an effective interface to 36-bit and wider devices.

For a complete list of 36-bit FIFOs please visit the Cypress web site at www.cypress.com. The web site also provides the latest data sheets, models, and any related documentation.