Interfacing Xilinx FPGAs to TI DSP Platforms Using the EMIF

Application Note

XAPP753 (v2.0.1) January 29, 2007

XILINX®

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
02/17/04	1.0	Initial Xilinx release.
03/31/04	1.1	Revised Figures, additions, and new information on reference design.
05/06/04	1.2	Revised title, added Platform.
06/15/05	2.0	Added Virtex-4 implementations. Converted to chapter format.
01/26/07	2.0.1	Changed link to reference designs.

Table of Contents

About This Guide

This application note provides several implementations that connect Xilinx FPGAs to a Texas Instruments Digital Signal Processor (DSP) platform using the External Memory Interface (EMIF).

Guide Contents

This manual contains the following chapters:

- Chapter 1, "EMIF Overview," provides an overview of the Texas Instruments EMIF.
- Chapter 2, "Virtex-II Series or Spartan-3 FPGA to EMIF Design," describes implementations connecting the TI TMSC6000 EMIF to Virtex™-II Series or Spartan™-3 FPGAs.
- Chapter 3, "Virtex-4 FPGA to EMIF Design," describes implementations connecting the TI TMS320C64x EMIF to Virtex-4 FPGAs.
- Chapter 4, "Reference Designs," provides the directory structure of the reference design and a link to the reference design files.
- Appendix A, "Virtex-4 ISERDES Sample Code," provides a sample code listing for a Virtex-4 implementation.
- Appendix B, "EMIF Register Field Descriptions," defines the TI DSP register fields.
- Appendix C, "Related References," provides links to related documents.

Additional Resources

To search the database of silicon and software questions and answers, or to create a technical support case in WebCase, see the Xilinx website at: http://www.xilinx.com/support.

Typographical Conventions

This document uses the following typographical conventions. An example illustrates each convention.

Convention	Meaning or Use	Example
Italic font	References to other documents	See the Virtex-4 *Configuration Guide* for more information.
	Emphasis in text	The address (F) is asserted *after* clock event 2.
Underlined Text	Indicates a link to a web page.	http://www.xilinx.com/virtex4

XILINX®

Chapter 1

EMIF Overview

This chapter briefly describes the functionality of the Texas Instruments DSP External Memory Interface (EMIF). This description covers only those elements needed for this application note and its related implementations.

For a complete overview of this interface and its functionality, refer to the appropriate TI datasheet (tms320c64*xx*.pdf, where *xx* is 15t, 16t, or 18).

EMIF

EMIF Signals

The TMSC64x EMIF is an enhanced version of the TMSC621x EMIF. It includes all the TMSC621x/TMSC671x EMIF features plus the following extra features:

- The data bus on EMIFA is either 64 bits or 32 bits wide.
- The data bus on EMIFB is 16 bits wide.
- The EMIF clocks ECLKOUTx are internally generated and based on the EMIF input clock. At device reset, one of the following three clocks can be configured as an EMIF input clock:
  - Internal CPU clock rate divide by four
  - Internal CPU clock rate divide by six
  - External ECLKIN
- Memories interfacing with the TMSC64x EMIF should operate off of ECLKOUTx (EMIF clock cycle).
  - The programmable synchronous memory interface controller's synchronous control pins replace the fixed SBSRAM control pins.
  - The PDT pin provides external-to-external transfer support.

Figure 1-1 shows the signals that make up the EMIFA and EMIFB interfaces, and Table 1-1 describes the signals. Signals for port A have the prefix "A" and signals for port B have the prefix "B". For practical reasons, all signal names in this document omit the EMIF port prefix.

All active-Low signals have a line over the signal name, such as $\overline{\text{SWE}}$.

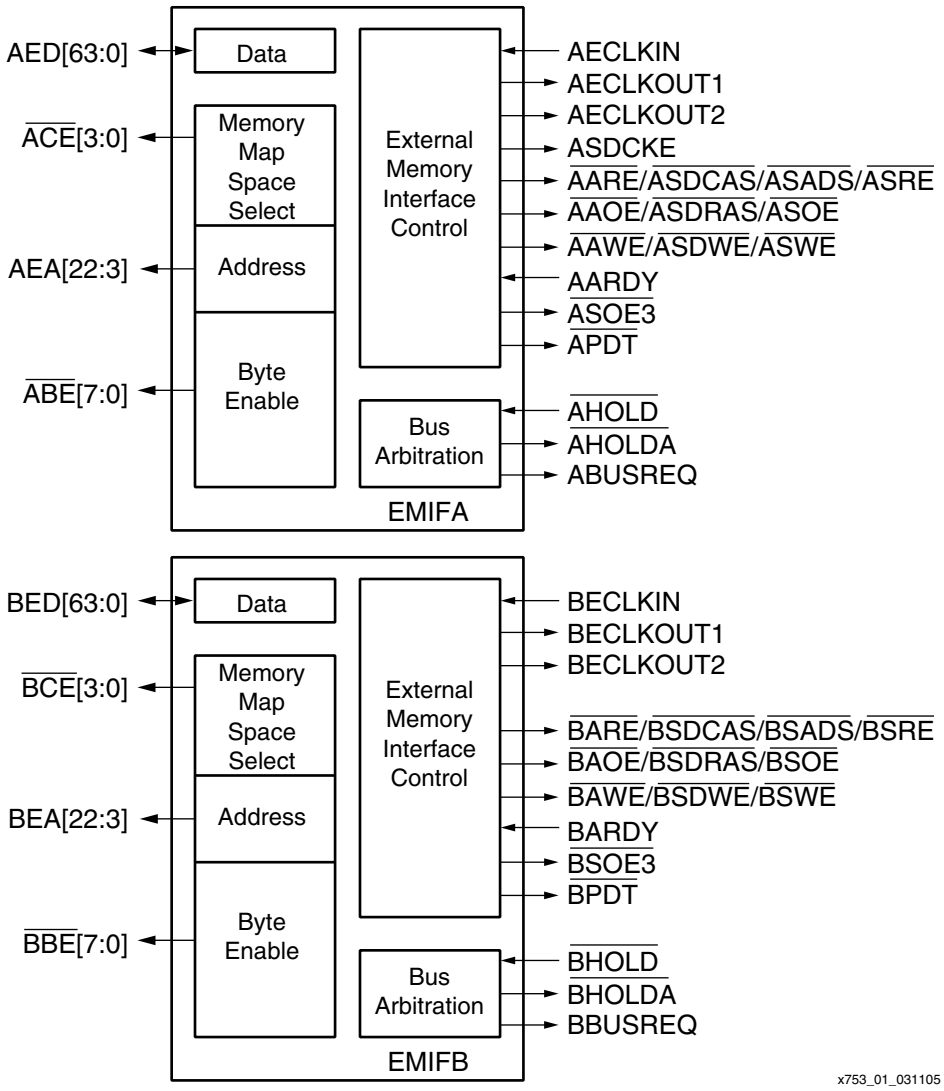


Figure 1-1: **EMIFA and EMIFB Signals**

Table 1-1: **EMIF Signal Descriptions**

Pin	I/O/Z	Description
CLKOUT4	O/Z	Clock output at ¼ the CPU clock rate
CLKOUT6	O/Z	Clock output at 1/6 the CPU clock rate
ECLKIN	I	EMIF clock input
ECLKOUT1	O/Z	EMIF output clock at ECLKIN, CPU/4, or CPU/6 rate
ECLKOUT2	O/Z	EMIF output clock at ECLKIN, CLKOUT4, or CLKOUT6 rate
ED[63:0]	I/O/Z	EMIFA 64-bit data bus14
ED[31:0]	I/O/Z	EMIFA 32-bit data bus
ED[15:0]	I/O/Z	EMIFB 16-bit data bus
EA[22:3]	O/Z	EMIFA address output

Table 1-1: **EMIF Signal Descriptions *(Continued)***

Pin	I/O/Z	Description
EA[20:1]	O/Z	EMIFB address output
$\overline{CE0}$	O/Z	Chip select for memory space 0
$\overline{CE1}$	O/Z	Chip select for memory space 1
$\overline{CE2}$	O/Z	Chip select for memory space 2
$\overline{CE3}$	O/Z	Chip select for memory space 3
$\overline{BE}$[7:0]	O/Z	EMIFA 64-bit byte enables. Byte enables are only active for the byte lane to which they correspond. $\overline{BE7}$ corresponds to ED[63:56] $\overline{BE6}$ corresponds to ED[55:48] $\overline{BE5}$ corresponds to ED[47:40] $\overline{BE4}$ corresponds to ED[39:32] $\overline{BE3}$ corresponds to ED[31:24] $\overline{BE2}$ corresponds to ED[23:16] $\overline{BE1}$ corresponds to ED[15:8] $\overline{BE0}$ corresponds to ED[7:0]
$\overline{BE}$[3:0]	O/Z	EMIFA 32-bit byte enables. Byte enables are only active for the byte lane to which they correspond. $\overline{BE3}$ corresponds to ED[31:24] $\overline{BE2}$ corresponds to ED[23:16] $\overline{BE1}$ corresponds to ED[15:8] $\overline{BE0}$ corresponds to ED[7:0]
$\overline{BE}$[1:0]	O/Z	EMIFB 16-bit byte enables. Byte enables are only active for the byte lane to which they correspond. $\overline{BE1}$ corresponds to ED[15:8] $\overline{BE0}$ corresponds to ED[7:0]
ARDY	I	Asynchronous ready input. Insert wait states for slow peripherals
$\overline{SOE3}$	O/Z	Synchronous output enable for $\overline{CE3}$
$\overline{AOE}$	O/Z	Asynchronous output enable
$\overline{SDRAS}$	O/Z	Row address strobe for DRAM memory
$\overline{SOE}$	O/Z	Synchronous output enable
$\overline{ARE}$	O/Z	Asynchronous read enable
$\overline{SDCAS}$	O/Z	Column address strobe for SDRAM memory
$\overline{SADS}$/$\overline{SRE}$	O/Z	Synchronous address strobe or read enable
$\overline{AWE}$	O/Z	Asynchronous write strobe
$\overline{SDWE}$	O/Z	Write enable for SDRAM
$\overline{SWE}$	O/Z	Synchronous write enable
$\overline{HOLD}$	I	External bus hold request
$\overline{HOLDA}$	O	External bus hold acknowledge
BUSREQ	O	Bus request
$\overline{PDT}$	O/Z	Peripheral data transfer
SDCKE	O/Z	ADRAM clock enable

EMIF Clocking

The external ECLKIN clock can be delivered by the FPGA. The EMIF can be clocked from an external clock, ECLKIN, or from the internal CPU (CPU/4 or CPU/6) clock. An output of the FPGA can be used to supply a quality clock for the EMIF. The modularity and frequency of the supplied clock can be changed easily, which is an extra advantage. Two outgoing clocks, for peripheral use, are generated from this reference clock.

- ECLKOUT1: a mirror of the EMIF input clock
- ECLKOUT2: a divided version of the EMIF clock

Timing parameters for the input and output clock are shown in Figure 1-2 and are described in Table 1-2.

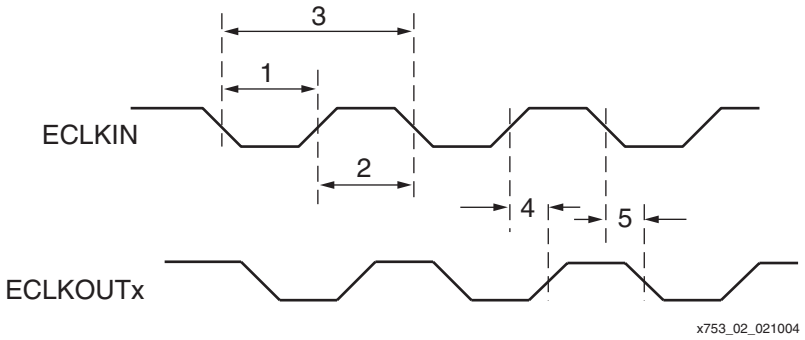


Figure 1-2: **ECLKIN, ECLKOUT1 Timing for EMIF Module**

Table 1-2: **ECLKIN, ECLKOUT1 Timing Parameters**

Parameter	Description	Min	Max	Units
1	ECLKIN Low	2.7		ns
2	ECLKIN High	2.7		ns
3	ECLKIN Period	6	16P	ns
4	ECLKIN High to ECLKOUT1 High ECLKIN High to ECLKOUT2 High	1 3	8 8	ns
5	ECLKIN High to ECLKOUT1 Low ECLKIN High to ECLKOUT2 Low	1 3	8 8	ns

Notes:

1. P = 1/CPU clock frequency (for 600 MHz, P = 1.67 ns).
2. The cycle-to-cycle jitter for ECLKIN is specified between 0 ps and 0.02 * (1/CPU).
3. The cycle-to-cycle jitter for the ECLKOUTx clocks is specified between 0 ps and 175 ps.

Byte-Lane Alignment

The EMIFA interface has the capability of interfacing to 8-, 16-, 32-, or 64-bit systems. The EMIFB interface port only supports 8-bit and 16-bit systems, as shown in Figure 1-3. External devices, memory as primary, are always right aligned to the ED[7:0] byte side of the bus. Endianess determines if bits ED[7:0] are accessed as bytes 0 (little endian) or as byte N (big endian where 2^N is the bus width). The different byte lanes can be selected through the application of active-Low byte-enable signals as shown in Table 1-1.

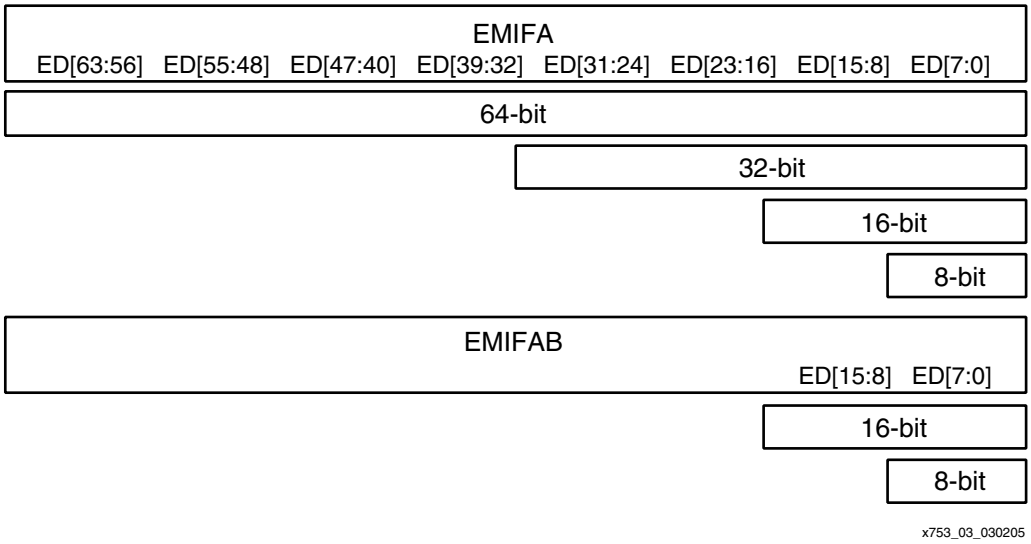


Figure 1-3: **Byte Alignment by Endianess**

EMIF Registers

To correctly operate the EMIF, some registers must be set and controlled. The different EMIF interfaces are memory mapped into the DSP memory space. Register addresses for EMIFA and EMIFB are listed in Table 1-3 and Table 1-4. For EMIFA, the base address is 0x0180_0000. For EMIFB, the base address is 0x01A8_0000.

Table 1-3: **EMIF Memory-Mapped Registers**

Address Offset	Register	Function
0x0000	GBLCTL	Global Control
0x0004	CECTL1	CE1 Space Control
0x0008	CECTL0	CE0 Space Control
0x0010	CECTL2	CE2 Space Control
0x0014	CECTL3	CE3 Space Control
0x0018	SDCTL	SDRAM Control
0x001C	SDTIM	SDRAM Refresh Control
0x0020	SDEXT	SDRAM Extension
0x0040	PDTCTL	Peripheral Device Transfer Control
0x0044	CESEC1	CE1 Space Secondary Control
0x0048	CESEC0	CE0 Space Secondary Control
0x0050	CESEC2	CE2 Space Secondary Control
0x0054	CESEC3	CE3 Space Secondary Control

Table 1-4: **EMIF CEx Selected Memory Space**

Memory-Mapped Block	Size	Address Range
EMIFB CE0	64M	0x6000_0000 – 0x63FF_FFFF
EMIFB CE1	64M	0x6400_0000 – 0x67FF_FFFF
EMIFB CE2	64M	0x6800_0000 – 0x6BFF_FFFF
EMIFB CE3	64M	0x6C00_0000 – 0x6FFF_FFFF
EMIFA CE0	256M	0x8000_0000 – 0x8FFF_FFFF
EMIFA CE1	256M	0x9000_0000 – 0x9FFF_FFFF
EMIFA CE2	256M	0xA000_0000 – 0xAFFF_FFFF
EMIFA CE3	256M	0xB000_0000 – 0xBFFF_FFFF

EMIF Control Registers

This section summarizes the EMIF Control Registers. Detailed descriptions of the register fields within these registers are located in Appendix B, "EMIF Register Field Descriptions."

CE Control Registers

The CECTL(3:0) registers correspond to the four CE memory spaces supported by the EMIF (Table 1-4). Figure 1-4 shows a control register setup. MTYPE is an important parameter as it defines the memory type for the corresponding memory space. When a synchronous memory type is indicated as MTYPE, the remaining register fields have no effect. For an asynchronous memory type, the register settings are active.

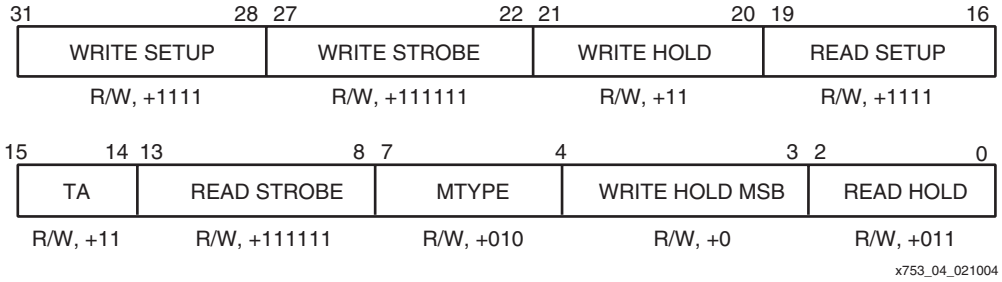


Figure 1-4: **EMIF CEx Space Control Register**

CE Secondary Control Registers

Synchronous memory can be controlled through the CE - Secondary Control Registers (CESEC). The CESEC(3:0) register fields are shown in Figure 1-5.

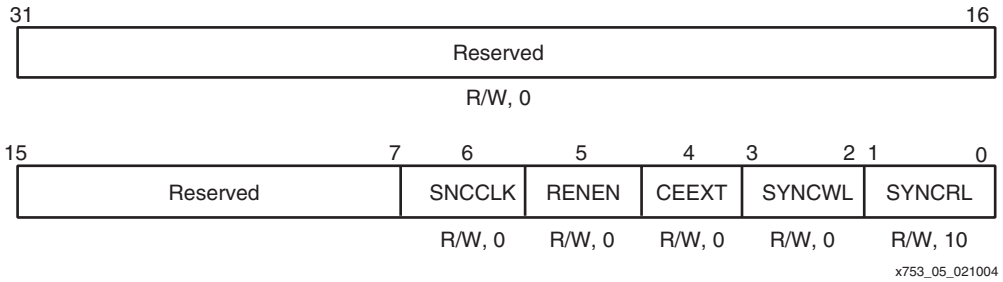


Figure 1-5: **EMIF CESEC Space Control Register**

Global Control Register

Figure 1-6 shows the register fields for the Global Control Register.

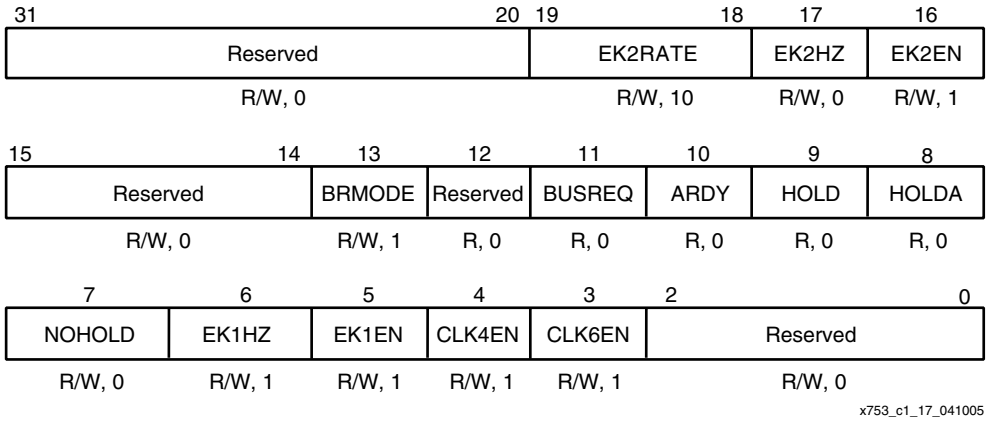


Figure 1-6: **Global Control Register**

Peripheral Device Transfer (PDT) Control Register

Figure 1-7 shows the register fields for the PDT Control Register.

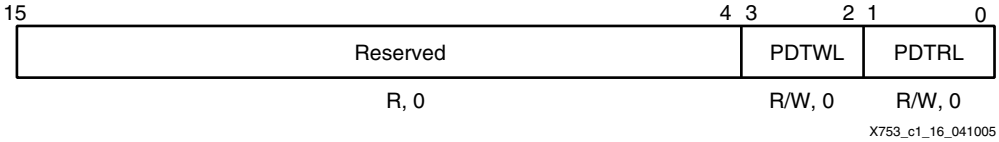
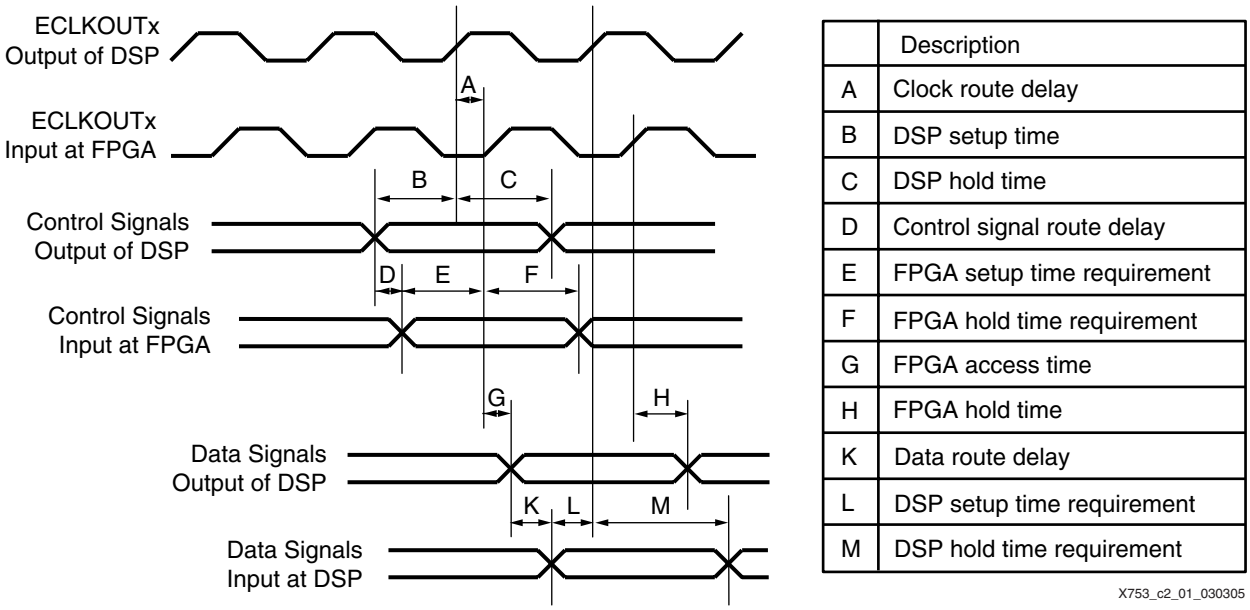


Figure 1-7: **PDT Control Register**

Board Level Parameters

This section emphasizes the effect of PCB traces on signals between components. Timing parameters in device data sheets do not include delays obtained through board routing. Good design practices always take these delays into account. Timing values can be altered by these delays. Texas Instruments and Xilinx provide IBIS models and guidelines to attain accurate timing analysis using a given setup.

Figure 1-8 shows general route delays as they are viewed between the DSP and the FPGA. From Virtex™-4 devices, the PCB trace delay differences can be compensated for by using the input delay line available in each IOB.



	Description
A	Clock route delay
B	DSP setup time
C	DSP hold time
D	Control signal route delay
E	FPGA setup time requirement
F	FPGA hold time requirement
G	FPGA access time
H	FPGA hold time
K	Data route delay
L	DSP setup time requirement
M	DSP hold time requirement

Figure 1-8: **Timing between EMIF, PCB, and FPGA**

Chapter 2

Virtex-II Series or Spartan-3 FPGA to EMIF Design

This chapter describes an interface between Virtex™-II, Virtex-II Pro, or Spartan™-3 devices to a Texas Instruments TMSC6000 DSP platform. The EMIF in the TMS DSP platform is used as the interface to the FPGA. Normally, the EMIF connects to different types of memory devices (such as SRAM, Flash RAM, DDR RAM). In this chapter, the EMIF connects to the FPGA, making the FPGA perform as a coprocessor, high-speed data processor, or high-speed data transfer interface. Texas Instruments has published EMIF application notes for memory designs.

The design interface example is a seamless connection to the FPGA block RAM. One side of the dual-port block RAM communicates with the DSP in Read/Write, FIFO, or Memory mode. The other side communicates with internal FPGA logic or processor(s).

FPGA Design

The flexibility of the FPGA makes it possible to create different designs, performing as different types of memory, with selectable bus widths (8-bit to 64-bit). Interfaces can be designed for the FPGA to work as synchronous or asynchronous standard memory, or as synchronous or asynchronous FIFOs.

Interfaces can be designed for the FPGA to interface synchronously or asynchronously with the EMIF. In synchronous mode, the ECLKOUTx clock is used to clock the FPGA interface logic. It is even possible to clock the entire FPGA from this clock.

An FPGA possesses an enormous amount of processing power using its logic functions, dedicated multipliers, PPC405 or MicroBlaze™ processors, and so forth. Thus the FPGA can serve as a coprocessor or a high-speed data processing and transfer device. The memory size of an FPGA is smaller than the possible addressable memory space in the TMSC64x type DSP. FPGA memory must be assembled using the available FPGA block RAM. The TMSC64x to FPGA interface described is of a FIFO type structure and an FPGA interface reacting as a memory block.

Block RAM

All recent Xilinx architectures have access to block memories. These 4 Kbit blocks in the Virtex, Virtex-E, and Spartan-II devices were increased in size to 18 Kbit blocks in the Virtex-II, Virtex-II Pro, and Spartan-3 devices. The blocks are fully synchronous, true dual-port memories.

The user can read from or write to each port independently (with the exception of simultaneous reads and writes to the same address). In addition, each port has a separate

clock, and the data widths for each port are independently programmable. Figure 2-1 shows a block diagram of the dual-port RAM blocks.

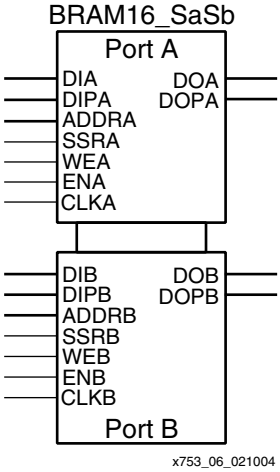


Figure 2-1: **Virtex-II Pro and Spartan-3 Block RAM**

The available block RAM in each device is described in the device datasheet. Table 2-1 shows an example using Virtex-II Pro devices. Table 2-2 shows the available memory sizes.

Table 2-1: **Block RAM per Virtex-II Pro Device**

Device Number	Blocks	Total Bits
XC2VP2	12	221184
XC2VP7	44	811008
XC2VP20	88	1622016
XC2VP40	192	3538944
XC2VP70	328	6045696

Table 2-2: **Block RAM Data Configuration Size**

Width	Depth	Address	Data	Parity
1	16384	ADDR(13:0)	D(0)	
2	8192	ADDR(12:0)	D(1:0)	
4	4096	ADDR(11:0)	D(3:0)	
9 (8 + 1)	2048	ADDR(10:0)	D(7:0)	DP(0)
18 (16 + 2)	1024	ADDR(9:0)	D(15:0)	DP(1:0)
36 (32 + 4)	512	ADDR(8:0)	D(31:0)	DP(3:0)

Block RAM FIFO

A FIFO made with FPGA logic and a number of block RAMs depends on the needed width and depth of the constructed FIFO. A FIFO can be built using the Xilinx CORE Generator™ tool or can manually be assembled in HDL.

Building the FIFO using the CORE Generator tool has the advantage that designs work and achieve high performance specifications. Designs built in HDL following designs specifications give designers the total freedom of design.

CORE Generator Tool

This software tool generates flexible, relationally placed macro (RPM) based FIFOs. It is necessary, from the TMSC64x DSP standpoint, to generate two RPM-based FIFOs: one read and one write FIFO. RPM-based designs are easy to multiply or place in a larger design. The combination of both is a full interface design to the DSP to pass blocks of data from the FPGA to and from the DSP. Table 2-3 shows the resources utilized per FIFO.

Table 2-3: **Device Utilization per FIFO**

Data Width	Depth	User Logic			Speed
		LUT	REG	Block RAMs	
32 bit	255	166	144	1	~200 MHz
64 bit	255	116	136	2	~180 MHz

TMSC64x to FPGA Interface Signals

For a FIFO interface, the standard TMSC64x EMIF FIFO interface scheme can be used. Table 2-4 summarizes the EMIF signals.

Table 2-4: **EMIF Signals**

Signal Name	Direction
$\overline{CE}$	DSP Output
$\overline{AOE}$	DSP Output
$\overline{AWE}$	DSP Output
$\overline{ARE}$	DSP Output
INTx	DSP Input
INTy	DSP Input
INTz	DSP Input
ED[63:0]	DSP Bidirectional

The FIFO requires a contiguous read clock and continuous write clock. These clocks are generated from the $\overline{ARE}$ and $\overline{AWE}$ signals, and are routed using the local clocking capabilities of the FPGA.

Because the DSP has 3.3V interfacing logic, the I/O bank(s) used for connecting to the EMIF must be specified as 3.3V V_{CCIO}. If this is not possible, level-shifting devices must be used. This introduces extra timing in the signal path between the two devices.

Design Examples

FIFO Interface

The standard EMIF FIFO setup is used for this design. Figure 2-2 shows a design example for Virtex-II Pro and Spartan-3 devices.

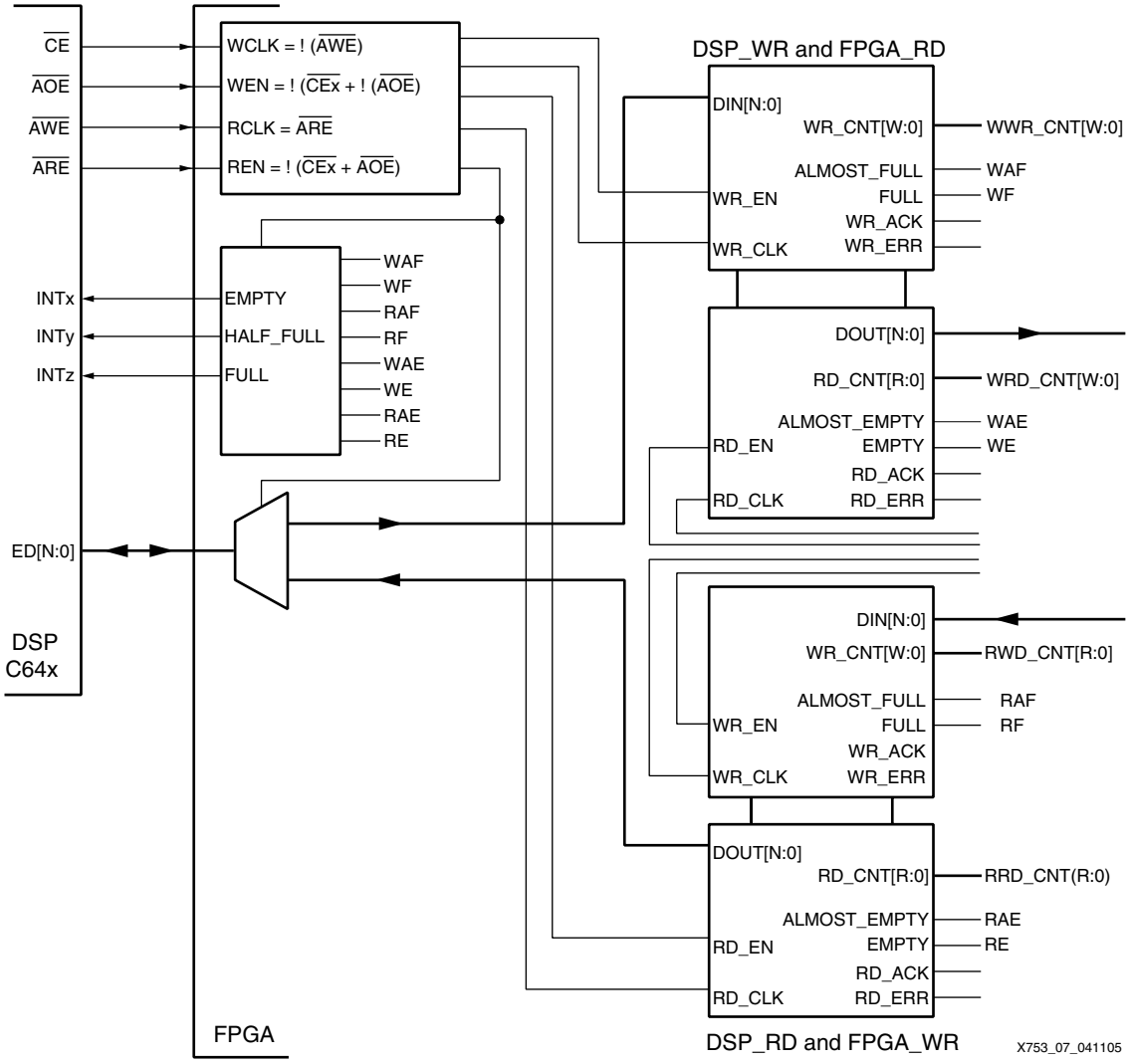


Figure 2-2: **FIFO-Based EMIF Design Example**

The data bank chip enable signal, $\overline{CE}$x, together with the asynchronous output enable, $\overline{AOE}$, is used to generate the enable signals for the write or read FIFO section of the design. This signal is also used to generate the enable signal for the data multiplexer and the flag selection logic.

The write and read FIFO clocks are routed via local clock routing directly to the FIFO clock inputs.

The RAMB16 components used in the FIFO are lined up along the I/O connected with the EMIF to substantially improve timing. Fast, rectangular interface designs can be constructed when the column of block RAM components are made the same size as the amount of I/O used.

Status Flag Signals

This design uses the normal FIFO flag outputs. For extra control over the flagging logic, whether for the DSP or for the FPGA side, use the write-counter and read-counter outputs. Signal AINIT, not shown in Figure 2-1, forces all flags to the active High state. On a first WR_CLK after AINIT is released, the FULL and ALMOST_FULL flags deassert Low. The same thing happens to the EMPTY and ALMOST_EMPTY flags on a first RD_CLK edge.

The ALMOST_EMPTY and ALMOST_FULL flags indicate that there is only one FIFO place left. User-defined FIFO flags can be constructed using the WR_COUNT and RD_COUNT outputs. The values of both counters do not reflect the exact position of the contents (address) of the FIFO. A clock latency of one clock cycle exists for both ports (clock domains) of the FIFO.

The outputs WR_COUNT and RD_COUNT are the upper bits of the internal FIFO address counter (delayed). When two bits are chosen for the output, the position of the FIFO can be determined as one fourth of the total size as shown in the Counter Decoding Example below. A finer position resolution can be obtained by making the counter wider.

Counter Decoding Example:

- COUNT[1:0] = 0b00 Indicates that the FIFO is less than ¼ full or empty.
- COUNT[1:0] = 0b01 Indicates that the FIFO is between ¼ and ½ full or empty.
- COUNT[1:0] = 0b10 Indicates that the FIFO is between ½ and ¾ full or empty.
- COUNT[1:0] = 0b11 Indicates that the FIFO is between ¾ and full or empty.

The reaction of the DSP to the different flags depends on the program running to read and write the FIFO in the FPGA.

EMIF Timing

Table 2-5 shows the EMIF timing parameters.

Table 2-5: **EMIF Timing Parameters**

Timing Parameter	Definition	Min	Max	Units
t_C	Cycle time, ECLKIN	6	16P	ns
t_{WH}	Pulse duration, ECLKOUT1 High	EH-0.7	EH+0.7	ns
t_{WL}	Pulse duration, ECLKOUT1 Low	EL-0.7	EL+0.7	ns
t_D	Output delay time, CLKOUT1 High to output active	1.3	4.9	ns
t_{OSUR}	Output setup time, select signals valid to $\overline{ARE}$	RS*E-1.5		ns

Table 2-5: **EMIF Timing Parameters *(Continued)***

Timing Parameter	Definition	Min	Max	Units
t_{OSUW}	Output setup time, select signals valid to $\overline{AWE}$	WS*E-1.7		ns
t_{OHR}	Output hold time, $\overline{ARE}$ High to signals invalid	RH*E-1.9		ns
t_{OHW}	Output hold time, $\overline{AWE}$ High to signals invalid	RW*E-1.8		ns
t_{SUR}	Setup time, read EDx valid before ECLKOUT1 High	2		ns
t_{HR}	Hold time, read EDx valid after ECKLOUT1 High	1.5		ns

Notes:

1. P = 1/CPU clock frequency in ns.
2. E = EMIF input clock period in ns.
3. EH = High period of E.
4. EL = Low period of E.
5. RS = Read Setup, value set in the CE space control registers.
6. WS = Write setup, value set in the CE space control registers.
7. RH = Read Hold, value set in the CE space control registers.
8. WH = Write Hold, value set in the CE space control registers.
9. ECLKOUT1 normally has the same period as the ECLKIN clock of the EMIF.

FPGA Timing

Table 2-6 shows a Virtex-II Pro -5 speed grade timing parameters.

Table 2-6: **FPGA Timing Parameters**

Timing Parameter	Definition	Min	Max	Units
T_{iopi}	I/O pad to IOB output		0.91	ns
T_{ioop}	IOB input to I/O pad		2.55	ns
T_{iotp}	3-state input to valid data on pad		2.48	ns
T_{ilo}	Four input LUT combinatorial delay		0.36	ns
T_{bcko}	Block RAM, clock to output		1.68	ns
T_{bpwh}	Block RAM, minimum pulse width High		1.50	ns
T_{bpwl}	Block RAM, minimum pulse width Low		1.50	ns

Figure 2-3 shows a write timing diagram.

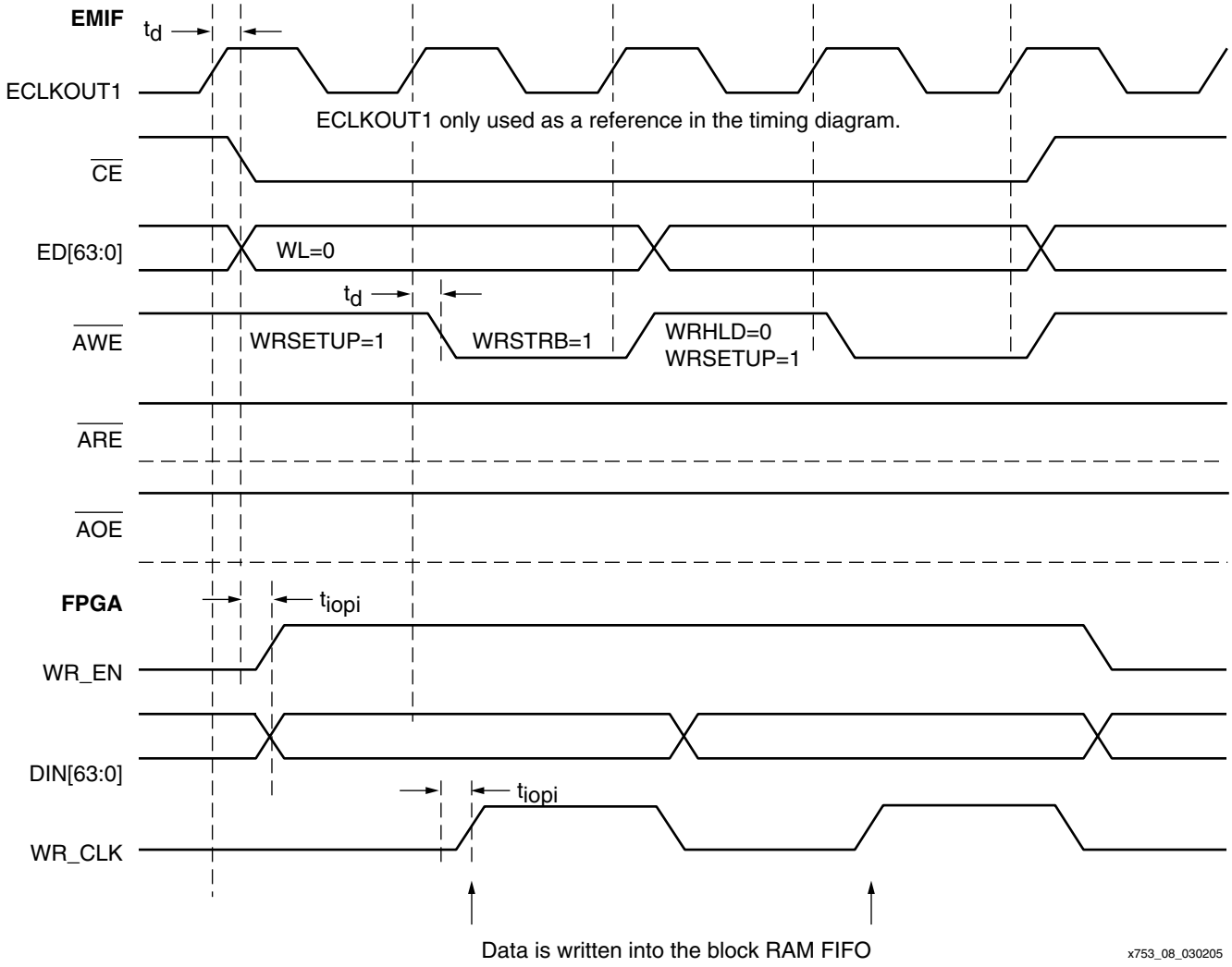


Figure 2-3: **Write Timing Diagram**

For the Write operation shown in Figure 2-3, the EMIF CE control register (CECTL) is set for when the asynchronous write enable ($\overline{AWE}$) is active one clock cycle after the CE is active. Data on the EMIF is also present.

The FPGA pad to IOB internal output timing for all I/O is equal. The time available from the IOB internal outputs to the rising edge of the WR_EN (inverted $\overline{AWE}$ signal) is one ECLKOUT1 cycle. In the minimum timing case, with the fastest ECLKOUT1, this time is approximately 6 ns.

The maximum speed at which the block RAM FIFO operates in Write mode is ½ ECLKOUT1.

Note: Route the WR_CLK over the local clock routing, or operate it with a stronger MAXDELAY timing constraint than for the other signals. All signals must get a timing delay constraint to keep them inside the timing range of the ECLKOUT1 clock.

Figure 2-4 shows a read timing diagram.

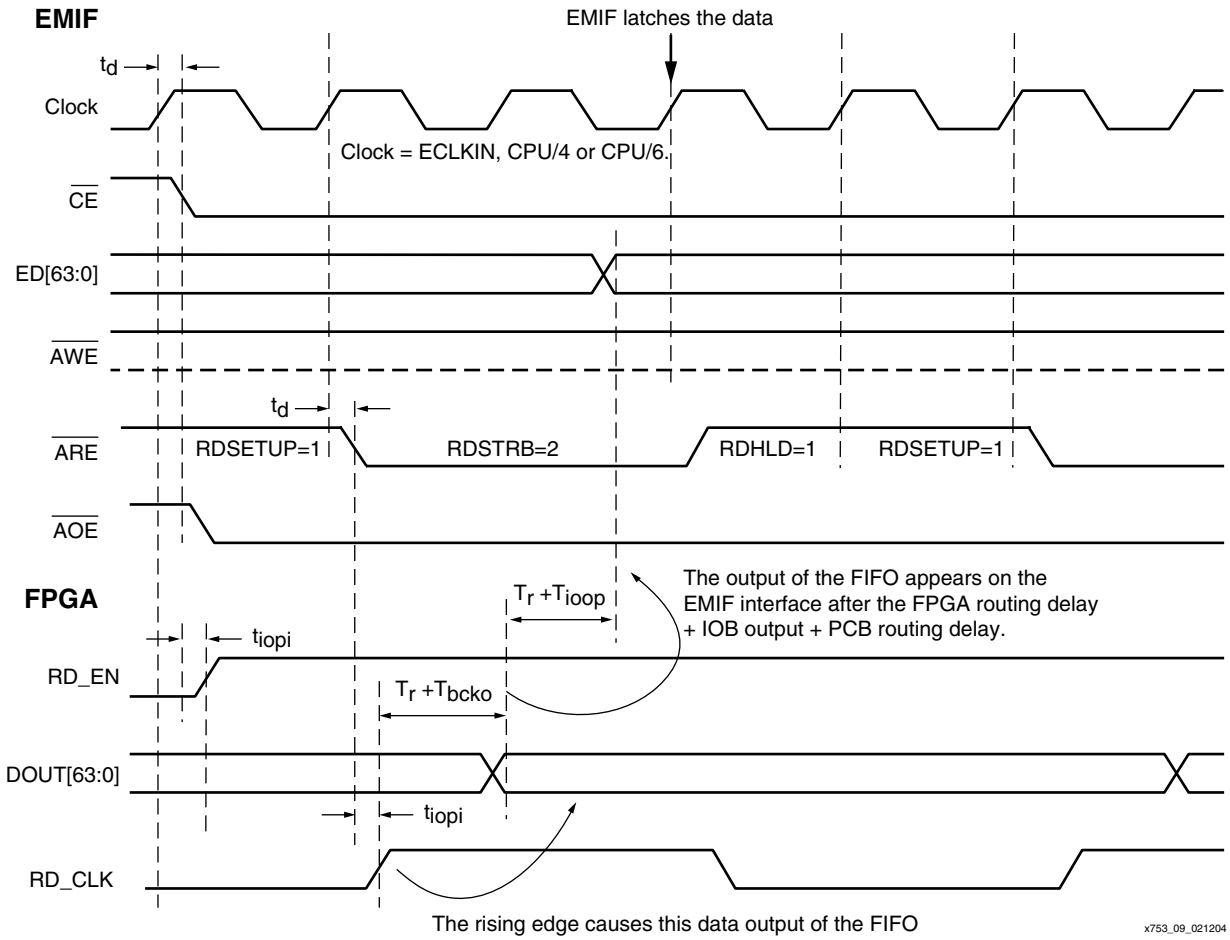


Figure 2-4: **Read Timing Diagram**

For the read operation in Figure 2-4, the CECTL register is set to have a one clock cycle setup delay. Two strobed clock cycles are needed and one hold cycle ends one read operation.

Two strobed clock cycles are needed to satisfy the output timing of the FPGA:

- It takes a routing delay (T_r) and a block RAM clock-to-output (T_{bcko}) delay to get the data out of the FIFO.
- It then takes a routing delay (T_r) and an IOB input-to-pad (T_{ioop}) to get the data out of the FPGA.
- A PCB routing delay brings the data to the EMIF.
- The sum of these delays is less than two EMIF clock cycles, 12 ns when ECLKOUT1 has a 6 ns period.

Block RAM Interface

A Block RAM interface can be used instead of a FIFO interface. This interface is simpler than the FIFO interface because all the logic to keep track of the different flags can be omitted. Only a small amount of decoding logic is needed for the EMIF control signals. This logic is limited to a few look-up tables (LUTs).

Figure 2-5 shows a 64-bit interface built from 8 block RAM components. This interface can be improved when constructed from 16 RAM16 components in a 4-bit configuration.

The byte enable ($\overline{BE}$) bits together with the control signals can be used to enable the different RAM blocks.

Parameters for the CESEC register are:

- SYNCRL = 0b01 – one cycle read latency
- SYNCWL = 0b00 – zero cycle write latency
- CEEXT = 0 – CE inactive when the final command is executed
- RENEN = 0 – $\overline{SADS}$ and $\overline{SRE}$ act as $\overline{SADS}$

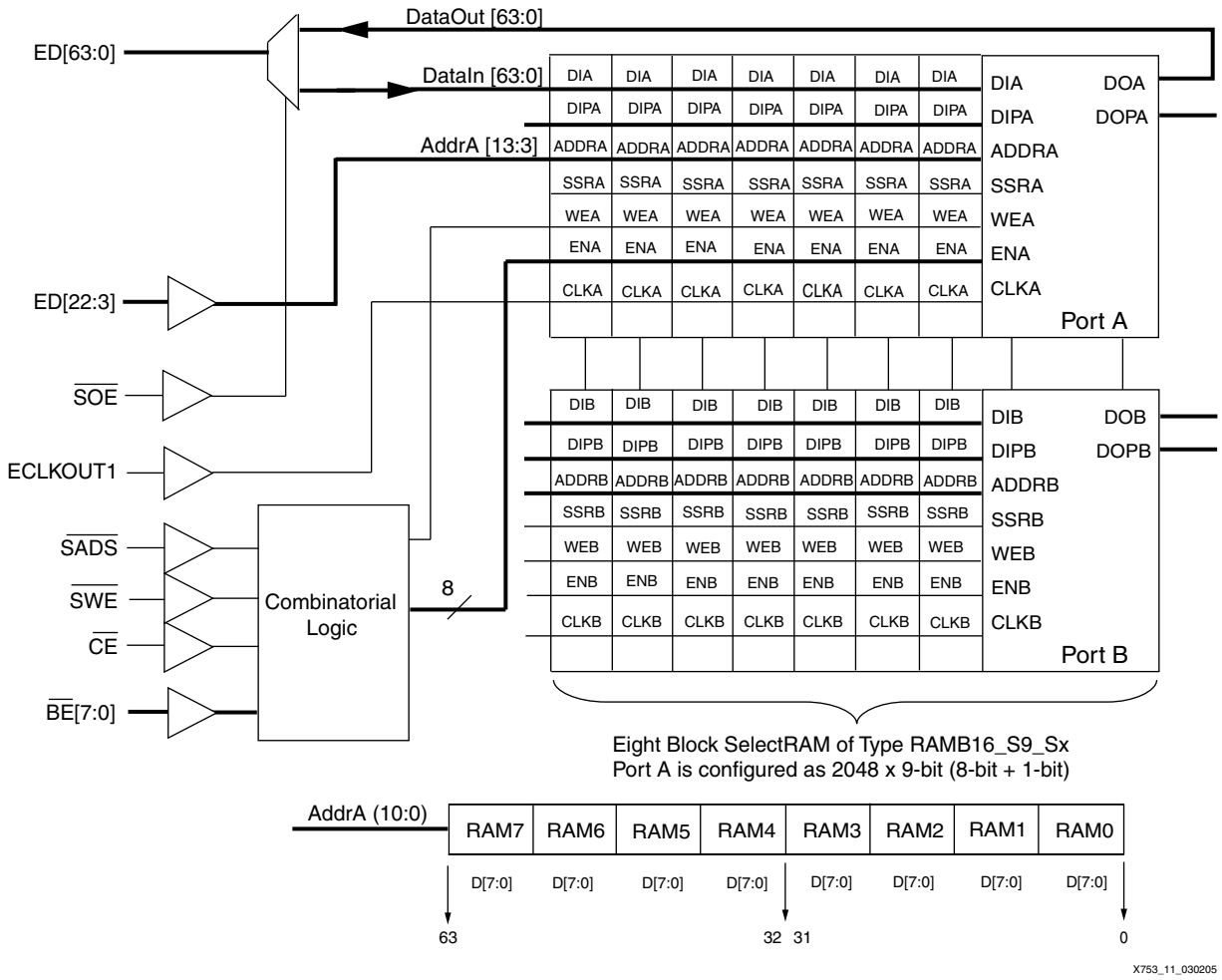


Figure 2-5: **64-Bit Memory-Based Interface Design Example**

Alternative Synchronous Interface

To obtain a faster data interface, it is possible to let all timing depend on the ECLKOUT1 clock. The TMSC64x by default generates a ECLKOUT1 clock derived from ECLKIN, CPU/4 clock, or CPU/6 clock.

The FPGA block RAM and additional logic use the ECLKOUTx clock as both a read clock and a write clock. The entire FPGA can be clocked from this ECLKOUTx clock. To accomplish this, the clock is fed into a digital clock manager (DCM). Figure 2-6 shows an easy-to-use, nearly glue-logic-free 32-bit synchronous design. The read and write timing diagrams for the 32-bit implementation are shown in Figure 2-7 and Figure 2-8, respectively.

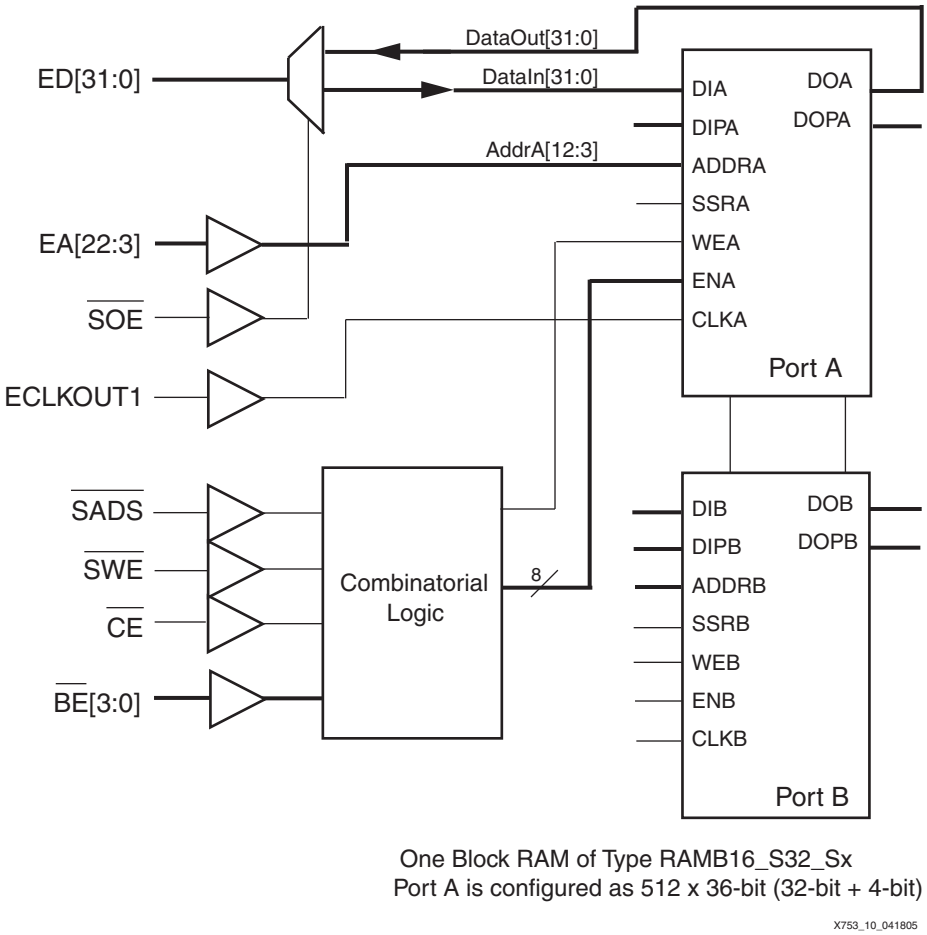


Figure 2-6: **32-Bit Memory-Based Interface Design Example**

Parameters for the CESEC register are:

- SYNCRL = 0b01 – one cycle read latency
- SYNCWL = 0b00 – zero cycle write latency
- CEEXT = 0 – CE inactive when the final command is executed
- RENEN = 0 – $\overline{\text{SADS}}$ and $\overline{\text{SRE}}$ act as $\overline{\text{SADS}}$

A useful addition to this design are flags indicating “data write to” or “data read from” to both the FPGA logic and DSP interrupts. DSP and FPGA software/hardware need to determine boundaries for read and write operations in the memory.

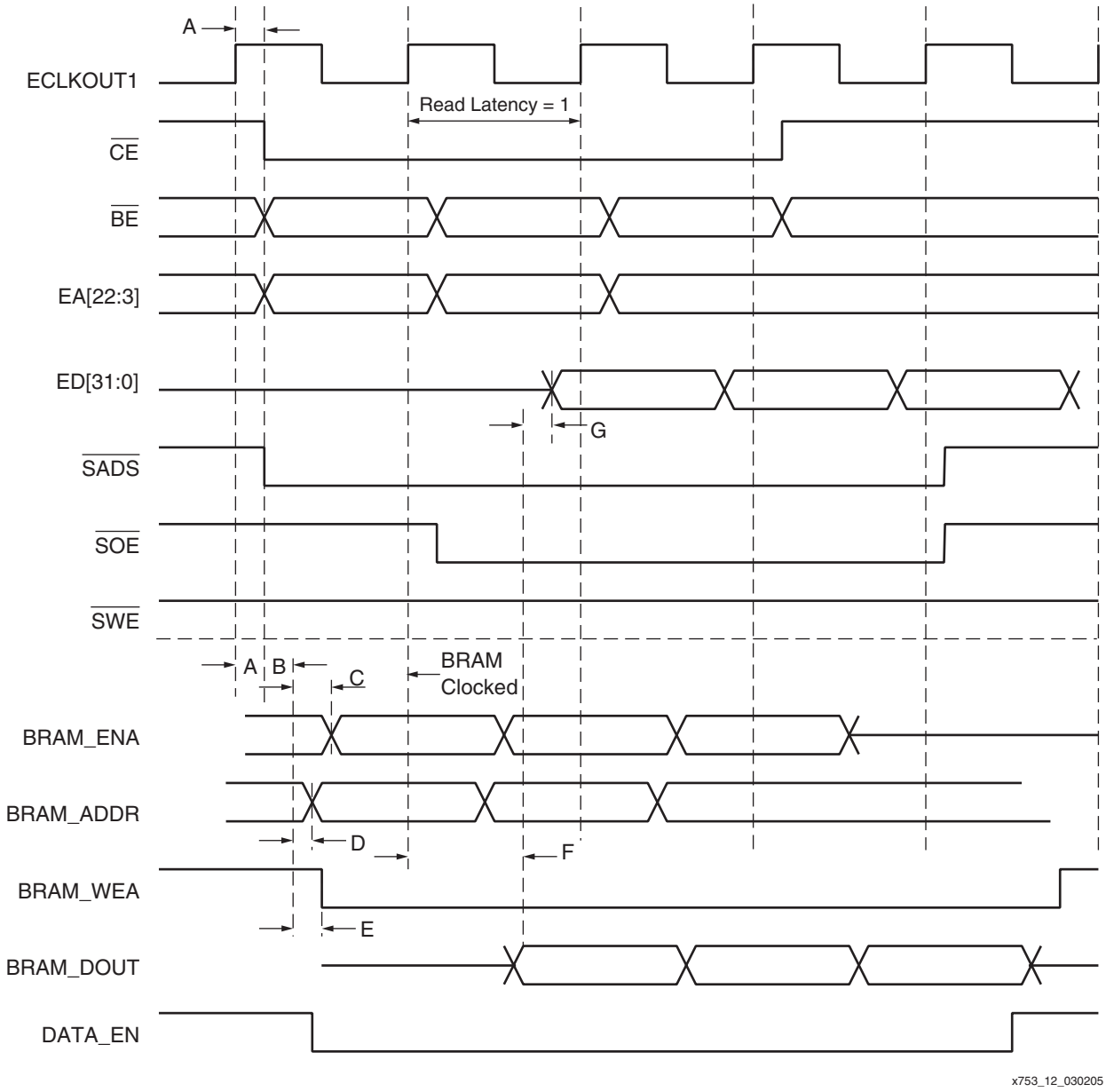


Figure 2-7: **32-Bit Read Timing Diagram**

Label A is the EMIF output delay time (CLKOUT1 high to output signal active). An additional delay must be added to get the different signals into the FPGA, or from the FPGA into the EMIF. These delays are a combination of PCB routing, FPGA I/O, logic, and routing delays.

The timing diagrams show this using the labels B, C, D, E, F, and G.

- Label B indicates the PCB routing delay.
- The EMIF CLKOUT1 clock is routed through a DCM (Digital Clock Manager) in the FPGA. The DCM lines the internal clock up with the external clock. If necessary, the DCM can shift the internally-generated clock over a specified phase difference to the external clock.

- The timing from the EMIF address bus to the FPGA internal block RAM can be calculated as $T_{pcb} + T_{iopi} + T_{net} + T_{back}$, and show as label D. This is when the address bus in the FPGA is routed directly from the input pads to the block RAM. The average delay is 2.653 ns.
- The combination of the $\overline{BE}$ and $\overline{CE}$ signals enables the FPGA RAM block. The delay of these signals from the FPGA pad to the block RAM enable input (label C) is 3.396 ns.
- The read/write strobe to the RAM is a combination of $\overline{SADS}$ and $\overline{SWE}$ signals, and the routing delay is 2.619 ns, as shown using label E.
- Data input and data output are controlled using the EMIF $\overline{SOE}$ signal. The delay of this signal in the FPGA is 1.845 ns.
- Data output from the Block SelectRAM happens on the next rising clock edge. The delay until the data appears at the FPGA pins is 4.699 ns. It then takes a PCB routing time to get the signals to the EMIF. This timing is shown as labels F and G.

Following are the calculations for the different delay parameters:

Address $T_{iopi} + T_{net} + T_{back}$

$\overline{BE}$ $T_{iopi} + T_{net} + T_{ilo} + T_{net} + T_{ilo} + T_{net} + T_{beck}$

$\overline{CE}$ $T_{iopi} + T_{net}$

$\overline{SADS}$ $T_{iopi} + T_{net} + T_{ilo} + T_{net} + T_{bwck}$

$\overline{SWE}$ $T_{iopi} + T_{net} + T_{ilo} + T_{net} + T_{ioop}$

Data Out $T_{bcko} + T_{net} + T_{ilo} + T_{net} + T_{iopi}$ $\overline{SOE}$

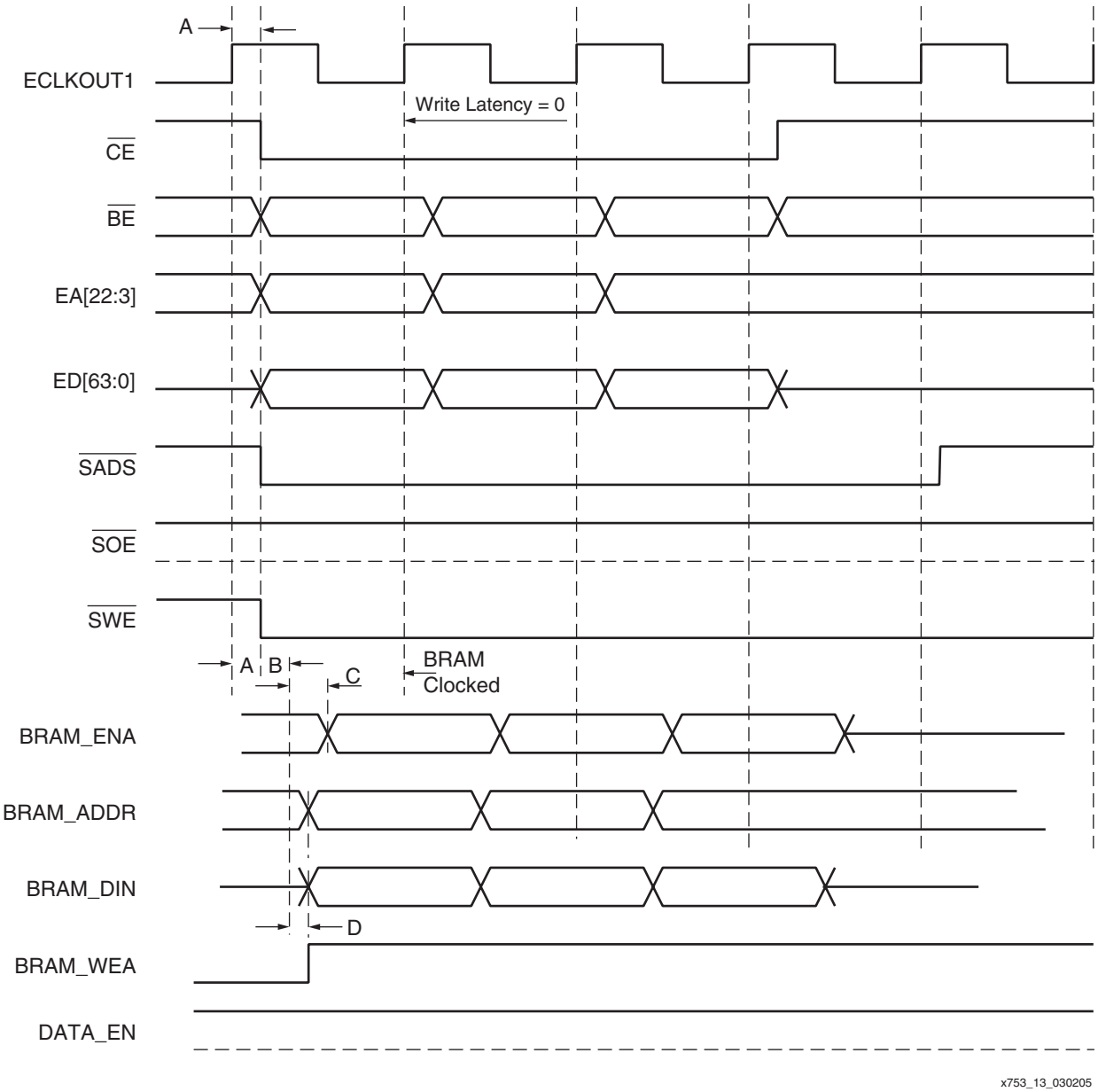


Figure 2-8: **32-Bit Write Timing Diagram**

An EMIF write operation to the FPGA block RAM shows nearly the same scenario:

$$\text{Data In} \quad T_{iopi} + T_{net} + T_{ilo}$$

$$\text{---------- } T_{net} \text{ ----------} + T_{bdck}$$

$$\overline{\text{SOE}} \quad T_{iopi} + T_{net} + T_{ilo}$$

Nearly the same description as the read timing applies for the write timing diagram. It is possible to write the block RAM devices without latency.

- Block RAM addressing and enable take the same amount of delay; labels A, B, and C are shown as for a read operation.
- Data from the EMIF into the FPGA until the block RAM has a timing of $T_{iopi} + T_{net} + T_{ilo} + T_{net} + T_{bdck}$, and shown as label D (2.620 ns).
- The data input is enabled through the use of the EMIF $\overline{SOE}$ signal. This signal enables the data input through a logic gate. The delay from the input pad until the look-up table output is 2.059 ns.
- The next rising ECLKOUT1 edge will latch the data into the DSP's EMIF registers.

The 32-bit EMIF is integrated into the FPGA using timing and placement constraints. This design is available at the link specified in Chapter 4, "Reference Designs." As an example, the placement of the design in the FPGA is shown in Figure 2-9.

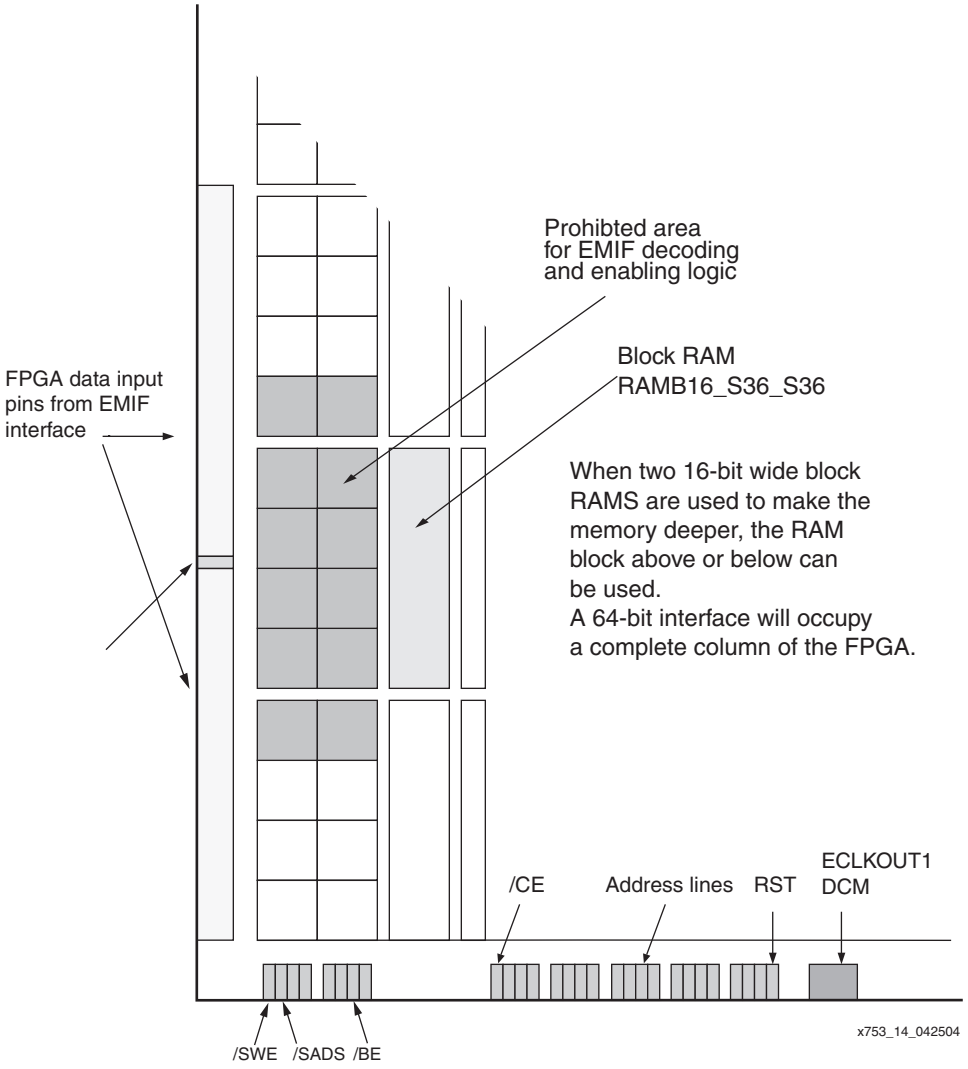


Figure 2-9: **32-bit EMIF Integration**

The reference design is synchronous. It is possible to design an asynchronous interface by using the ARDY signal. ARDY is an input to the EMIF port and is used as a “go” flag to the EMIF.

If an EMIF read operation is assumed:

- The EMIF provides address and control signals to the connected logic.
- The external logic in the FPGA holds ARDY low while accepting and processing the incoming data.
- Once the FPGA logic is finished accepting the incoming data, ARDY is forced high, indicating to the EMIF that it can continue with next process steps.

This way it is possible to stretch the EMIF bus cycle over several ECLKOUT1 clock cycles.

Conclusion

Interfacing the Texas Instruments TMSC64x DSP Platform to the Virtex-II Pro or Spartan-3 FPGAs is an easy and straightforward process. It is possible to work with standard FIFO or memory devices by using the Xilinx ISE tools. Due to the flexible timing parameters of the EMIF, very little FPGA logic is needed. This way, with a minimal design effort, the FPGA can be used as a DSP coprocessor, or perform as a high-speed data processing or high-speed data bridge device.

Chapter 3

Virtex-4 FPGA to EMIF Design

This chapter describes the connection of a Virtex-4 FPGA to the EMIF port of a TMS320C64x DSP. Because the enhanced Virtex-4 I/O structure is completely different from earlier Virtex-II series structures, new and faster FPGA interfaces can be designed to connect to the DSP EMIF.

This chapter describes several interfaces that use different combinations of dual-port block RAMs (RAMB16s) and FIFOs (FIFO16s). These interfaces do not use the Virtex-4 enhanced I/O features.

In addition, this chapter describes how to use the serializer and deserializer interfaces available in each I/O block. The Virtex-4 ISERDES and OSERDES structure interfaces are more flexible and powerful than in the Virtex-II series, providing new methods to designing external devices. Previous interface design methods that required a substantial amount of logic are replaced entirely using existing Virtex-4 I/O blocks and enhanced memory.

Virtex-4 IOB

In contrast with the IOB setup of the previous Virtex FPGA families, the Virtex-4 I/O block is split into three different parts. The basic IOB contains only the input and output (3-state) buffer. The rest of the IOB is split into ISERDES and OSERDES complex blocks. Figure 3-1 shows a Virtex-4 IOB. The *Virtex-4 User Guide* describes the details of the ISERDES and OSERDES blocks.

The IOB is used in its most simple format in "FPGA Interface," page 32, that of input and 3-state output. Some advanced interfaces are described in "Virtex-4 IOB with ISERDES and OSERDES Functionalities," page 52 using the ISERDES and OSERDES functions of the Virtex-4 IOB.

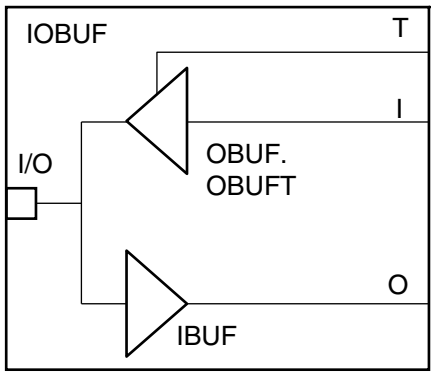


Figure 3-1: **Virtex-4 IOB**

FPGA Interface

Block RAM Used as Memory (Example 1)

Figure 3-2 shows an example configuration, where the block RAM is used as memory. This interface uses the Virtex-4 I/O as a simple input and 3-state output buffer.

This interface connects the DSP and the FPGA with a minimum of logic to obtain maximum throughput of data. The EMIF uses the FPGA as a memory device with a 32-bit by 4096 configuration.

To keep the connections between the FPGA I/O pads as short as possible, different RAMB16 blocks are used, resulting in a relatively large amount of memory available to the user of the interface.

Constraining the interface into an I/O bank gives direct and easy access to a set of RAMB16 dual-port block RAMs, where port A (B) is used as a DSP EMIF access port (read/write) and port B (A) is used as an FPGA access port for read and/or write operations. The memory access port of the DSP is connected to the Virtex-4 PPC405 processor through the CoreConnect bus or DSOCM bus.

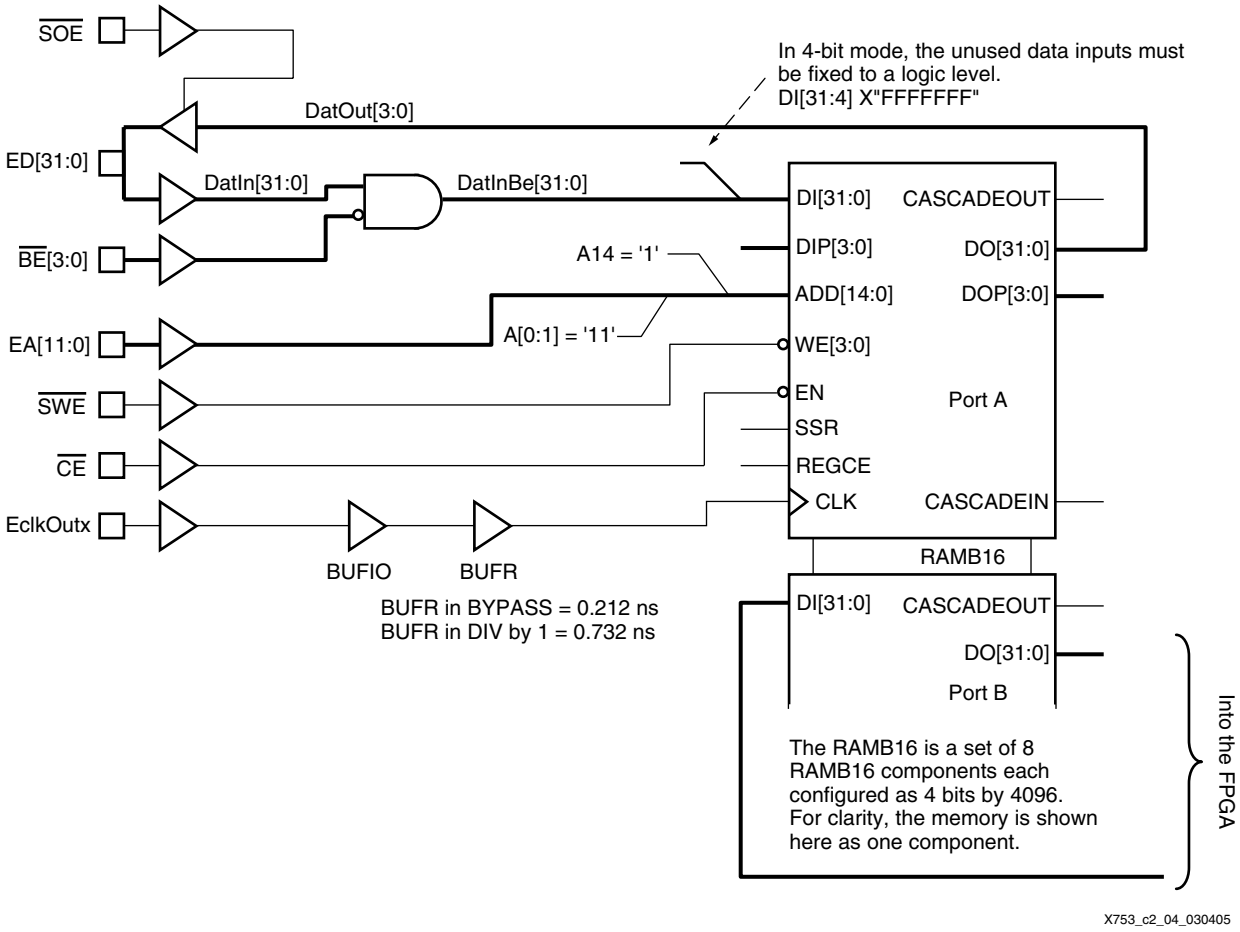


Figure 3-2: **EMIF – FPGA Memory Interface (Example 1)**

The interface I/O is LOC'ed into an I/O bank using the following syntax:

```
CONFIG PROHIBIT = "pad_X";
INST "Dsp*" LOC = BANK<number_of_I/O_bank>;
CONFIG PROHIBIT = "Pad_Y";
```

Each I/O connection to the EMIF starts the same (*Dsp…*), allowing the use of the "*" wildcard operator to constrain all I/Os at once.

With the "CONFIG PROHIBIT" operator, the entire I/O for the EMIF is placed in the middle of the I/O bank.

In this example, the interface I/O is constrained into bank 9 of an XC4VLX25 as shown in Figure 3-3.

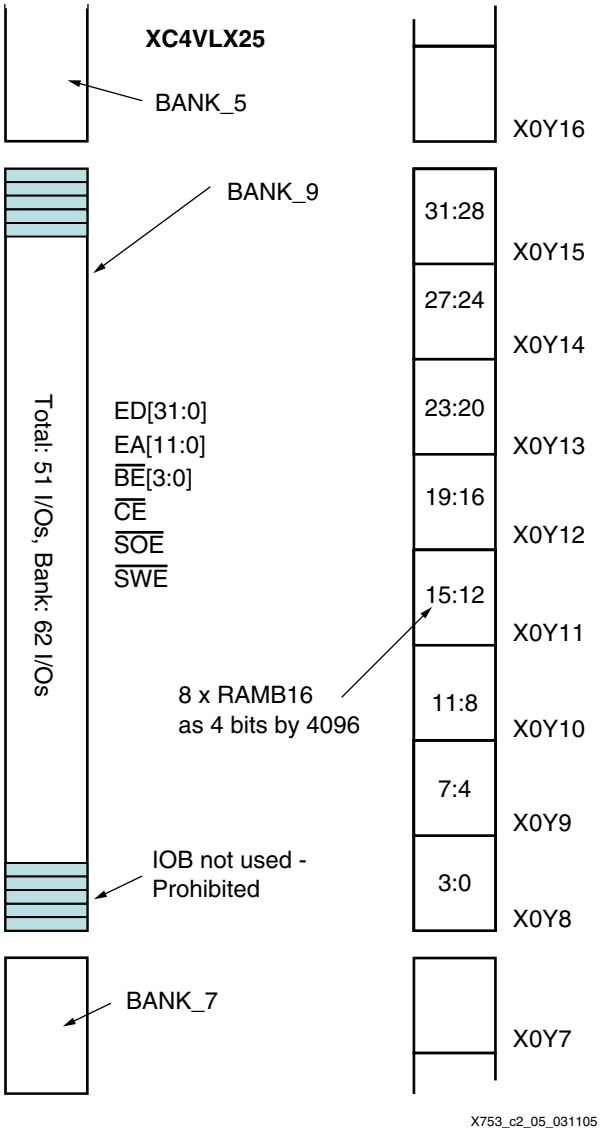


Figure 3-3: **FPGA Interface Setup**

Figure 3-4 shows the EMIF write timing waveforms, and Figure 3-5 shows the EMIF read timing waveforms.

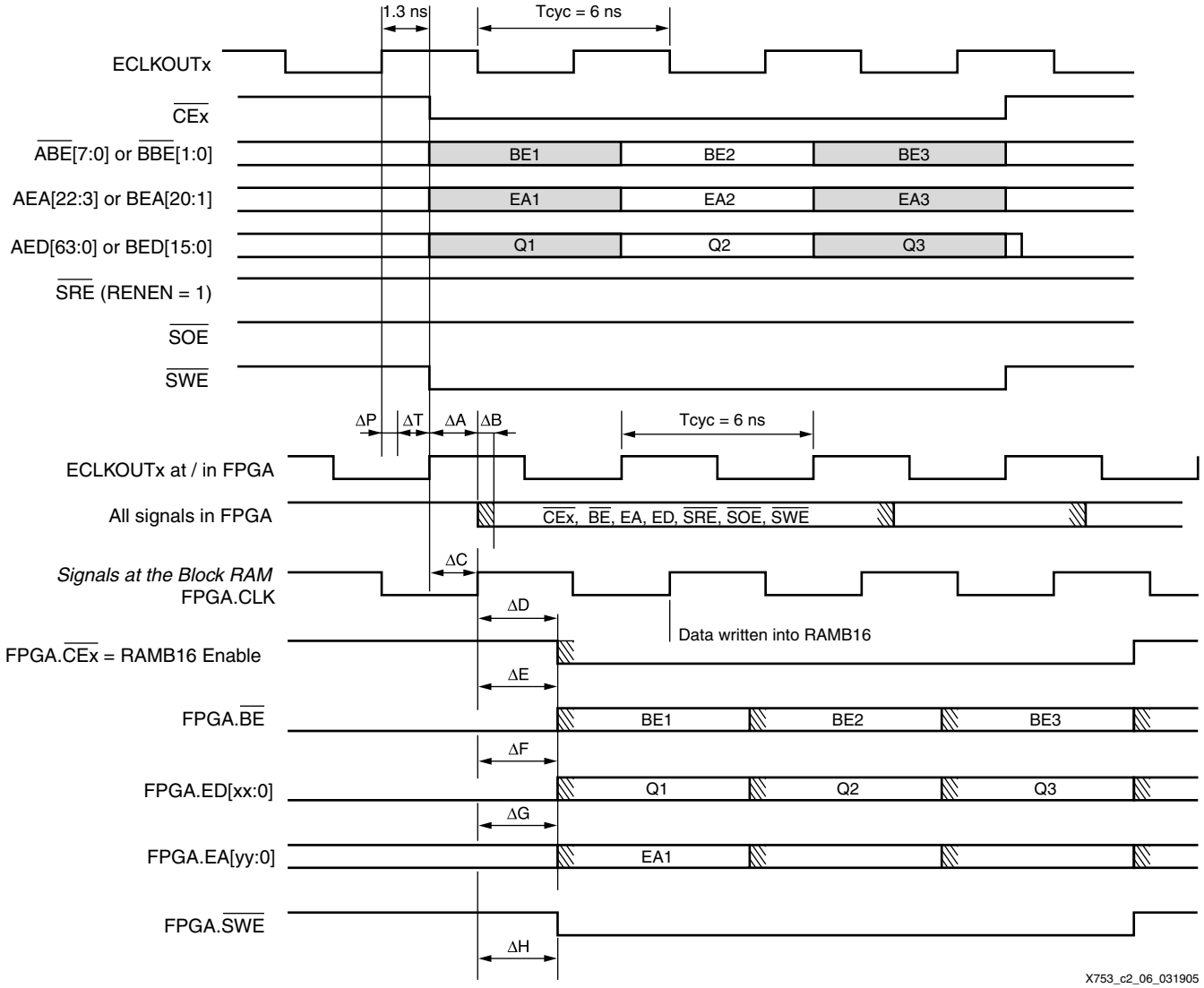


Figure 3-4: **EMIF Write Operation and FPGA Timing**

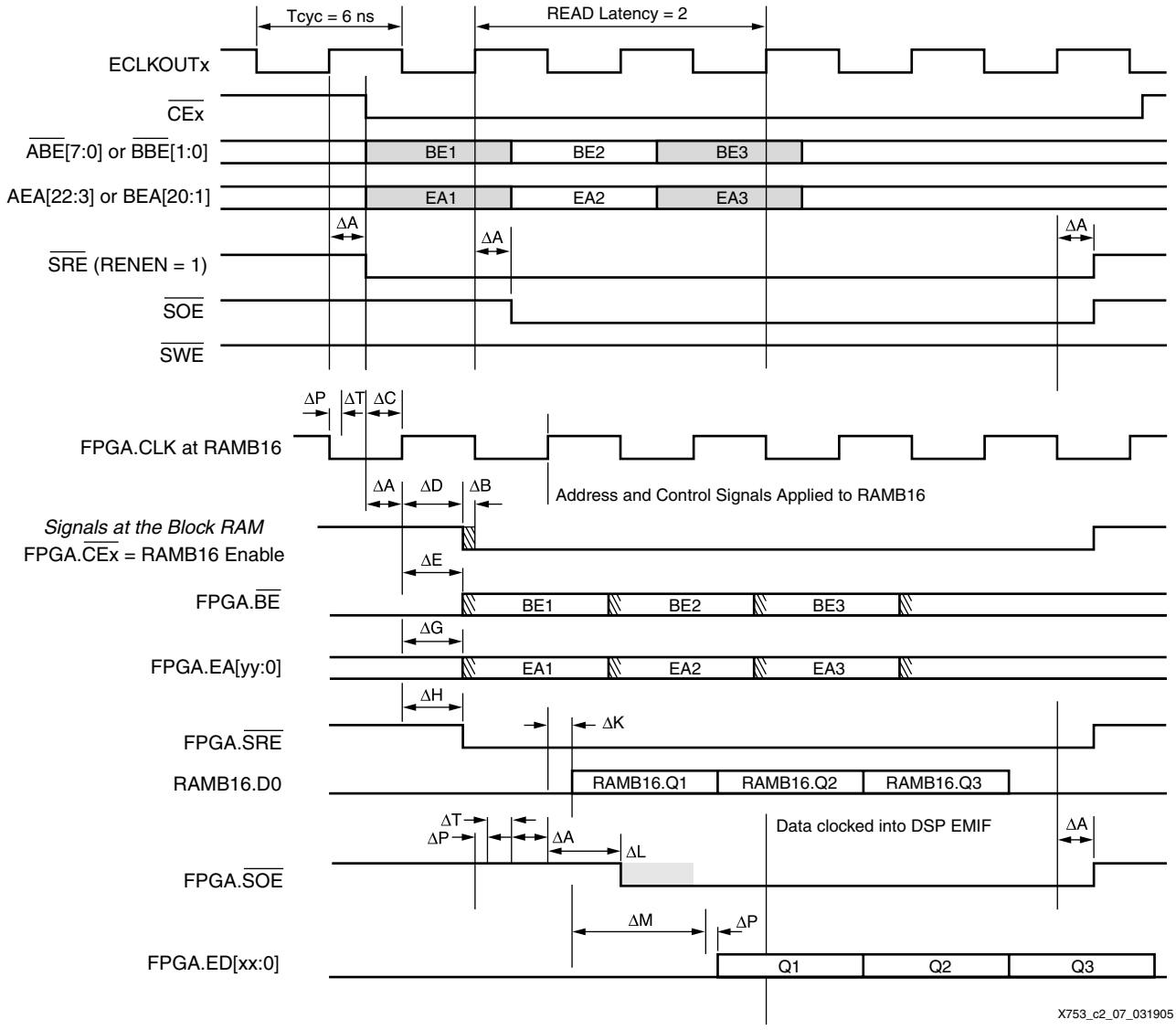


Figure 3-5: **EMIF Read Operation and FPGA Timing**

Table 3-1 defines the timing parameters shown in Figure 3-4 and Figure 3-5.

Table 3-1: **Timing Parameter Descriptions for Example 1**

Parameter	Description
DSP	
ΔA	Minimum CLKOUTx to signal valid delay of the DSP (1.3 ns).
PCB	
ΔP	PCB trace delay between DSP and FPGA. This signal is the travel time between the pins of both devices. A trace length of 100 mm requires ~600 ps. For an EMIF read operation, this delay must be counted twice – once to address the FPGA and once to collect the data from the FPGA.
ΔB	Allowed, calculated time for signal setup at the FPGA due to PCB trace length differences. This value is ~ 60 ps. This delay can be omitted when the PCB is properly laid out.
FPGA Inputs(1)	
ΔT	FPGA input delay (Tiopi = 0.954 ns). This delay is the same for all FPGA inputs. It is the time between the package pad and the output of the IOB input buffer (IBUF.O).
ΔC	Internal FPGA delay of the clock. From the IBUF.O to the input of the RAMB16 block. This delay is built as: net + Tbufiocko_O + net + Tbrcko_O_BYP + net ~0.400 ns + 0.0 ns + 0.101 ns + 0.212 ns + ~1 ns = ~1.7 ns
ΔD	Internal FPGA delay from the IBUF.O of the CE pin to the RAMB16. net + Trcck_ENA = ~1.8 ns + 0.403 ns = ~2.2 ns
ΔE	Internal FPGA delay from the IBUF.O of the BE pins to the RAMB16. net + Tilo + net + Trdck_DIA ~1.2 ns + 0.194 ns + ~0.8 ns + 0.115 ns = ~2.3 ns
ΔF	IBUF.O of the ED data pins to the RAMB16 memory. net + Tilo + net + Trdck_DIA ~1.1 ns + 0.194 ns + ~0.9 ns + 0.115 ns = ~2.3 ns
ΔG	Internal address routing and setup delay. net + Trcck_ADDRA ~2 ns + 0.311 ns = ~2.3 ns
ΔH	Internal FPGA routing and setup for the WEA ($\overline{SWE}$) signal. net + Trcck_WEA ~1.9 ns + 0.630 ns = ~2.53 ns
ΔK	RAMB16 clock to data out on the O pins (Tcrko_DO = 920 ps).
ΔL	Internal $\overline{SOE}$ to 3-state buffer of IOB. net + Totq + net 1.9 ns + 0.799 ns + 0.120 ns = ~2.8 ns

Table 3-1: **Timing Parameter Descriptions for Example 1 *(Continued)***

Parameter	Description
FPGA Output	
ΔM	Data from RAMB16.O outputs to the output package pad. net + Tioop ~2.1 ns + 2.961 ns = ~5.1 ns

Notes:

1. To calculate the delay between the DSP to the on-die IOB, add Tiopi = 0.954 ns (the value shown as ΔT in the timing diagrams) plus the package flight delay to the PCB delay.

To achieve the timing and performance listed in Figure 3-4, Figure 3-5, and Table 3-1, the CESEC and CECTL registers are configured as follows:

- CESEC register:
  - SYNCRL is set to 2 (read latency).
  - SYNCWL is set to 0 (write latency).
  - RENEN is set to 1 ($\overline{\text{SADS}}$/$\overline{\text{SRE}}$ act as $\overline{\text{SRE}}$).

    When RENEN = 1, $\overline{\text{SRE}}$ and $\overline{\text{SWE}}$ are opposites, and therefore only one of the two signals can be used to access the Virtex-4 block RAM. In this case, $\overline{\text{SWE}}$ is used.
  - CEEXT set to 1 ($\overline{\text{CE}}$ is active when $\overline{\text{SOE}}$ is active).
- CECTL register:
  - MTYPE is set to 0xE (64-bit synchronous interface for EMIFA).
  - MTYPE is set to 0x4 (32-bit synchronous interface for EMIFA).
  - MTYPE is set to 0xB (16-bit synchronous interface for EMIFA/EMIFB).

Block RAM Used as Memory with Front-Side Flip-Flops (Example 2)

Figure 3-6 shows an example configuration, where the block RAM is used as memory with front-side flip-flops. This example registers all EMIF signals in the FPGA before they are used. The registers in the ISERDES (ILOGIC) are used. When the DSP EMIF is writing to the FPGA interface, timing is related to the registers (flip-flops) in the FPGA inputs. The memory (RAMB16) in the Virtex-4 FPGA is accessed and written in the following clock cycle.

To not lose an entire clock cycle (when the timing of the FPGA allows it), the RAMB16 memory component in the Virtex-4 FPGA can be accessed at the opposite edge of the input registers.

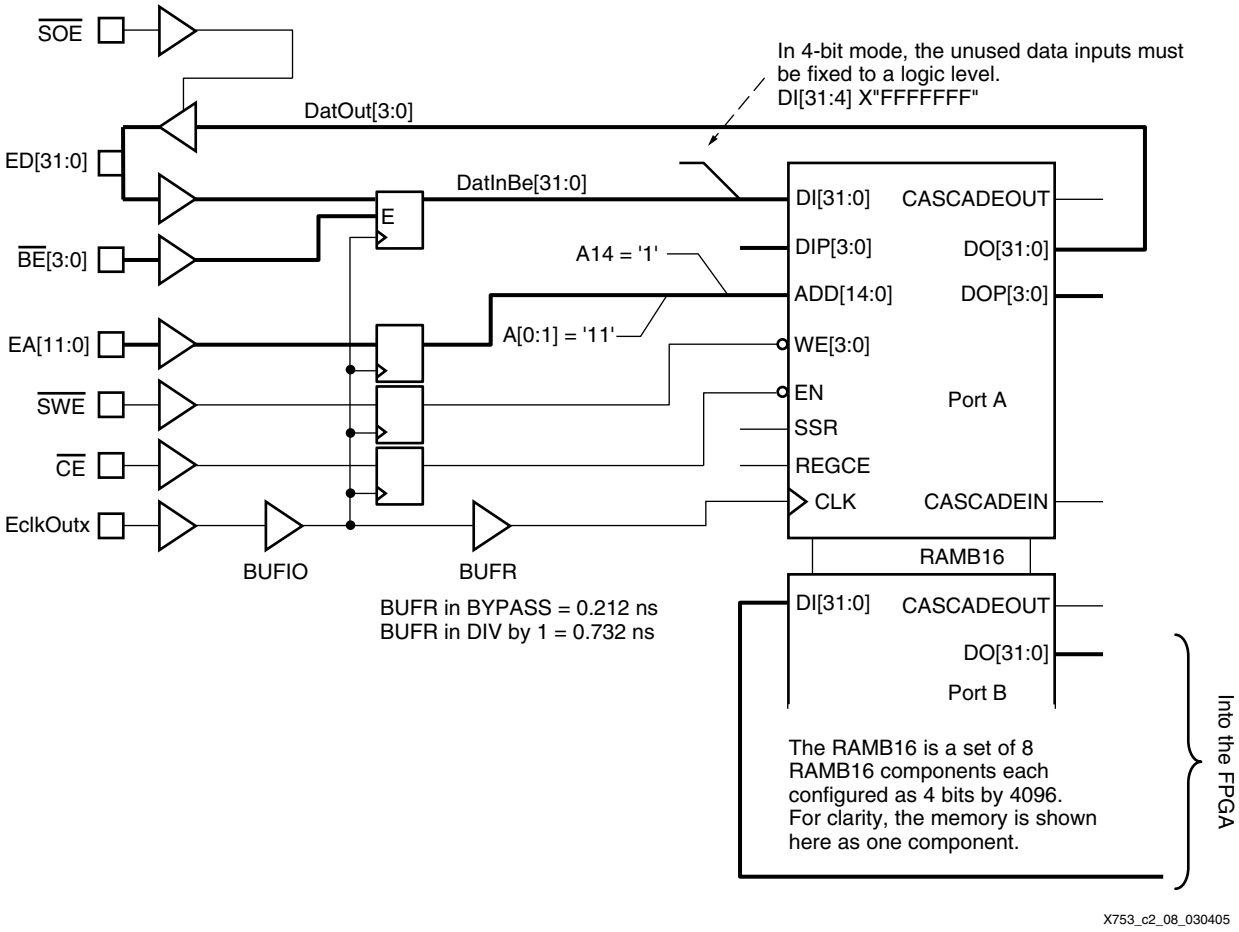


Figure 3-6: **EMIF Using I/O Registers (Example 2)**

This design uses the clock capable I/O for the ECLKOUTx clock of the DSP.

The input flip-flops are clocked with the output of the BUFIO, which is a dedicated buffer for clocking I/O flip-flops. The output of the BUFIO is the source for the BUFR clock buffer. The BUFR clock buffer clocks all logic in the region where it resides. In this case, it clocks the RAMB16 memory blocks.

Note: BUFR is a dedicated local clock, capable of clocking all logic in the region in which it is placed and the region above and under this region.

The skew between the clock for the input registers and the clock of the memory is ~180 ps. The input registers are clocked before the memory blocks. Using this knowledge, the input flip-flops can be clocked on the rising edge of the IntClk clock, and the RAMB16 components can be clocked on the falling edge of the IntClkDiv clock.

Figure 3-7 shows the ECLKOUTx clock setup and timing.

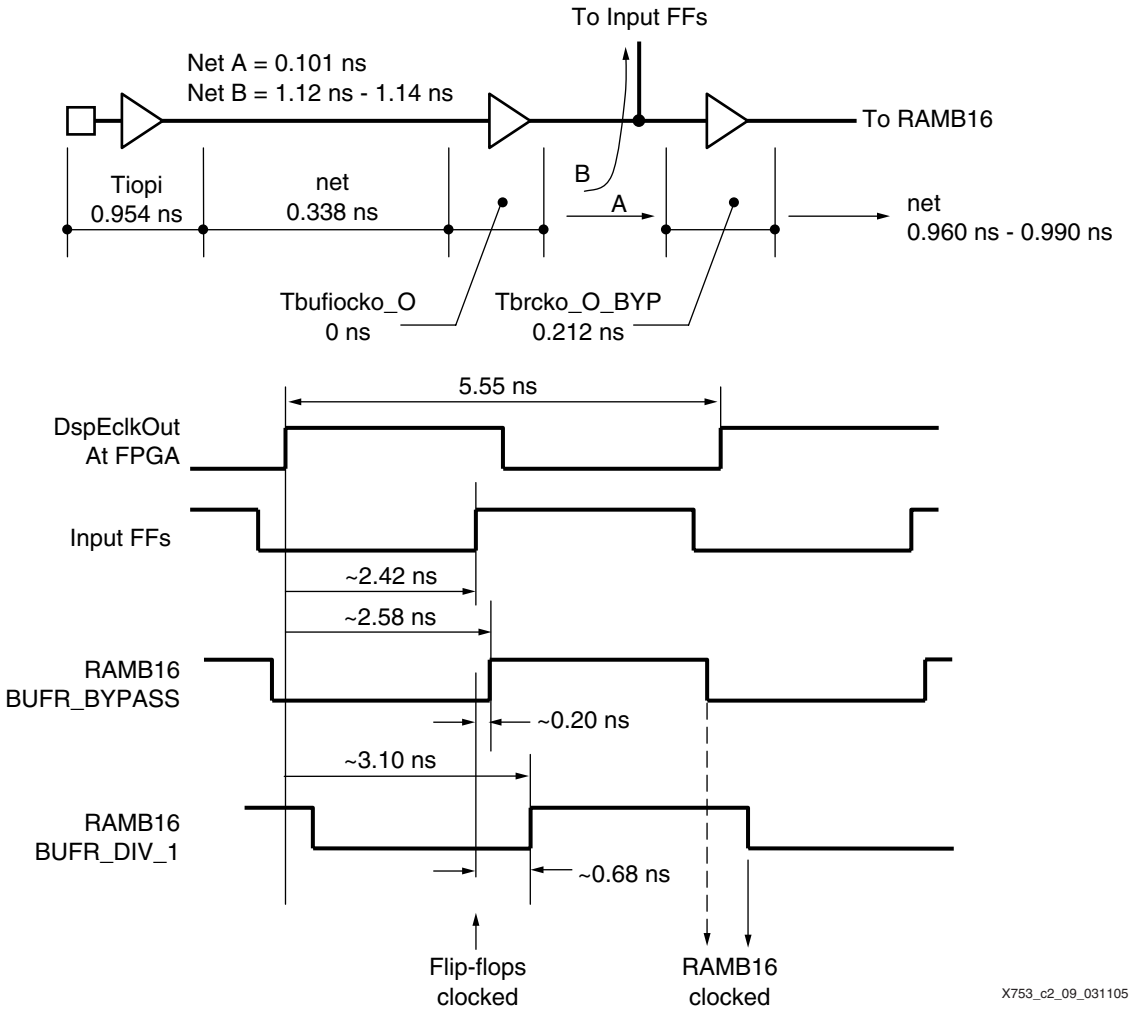


Figure 3-7: **Clock Timing Diagram**

The following assumptions are made for the timing shown in Figure 3-7:

- The ECLKOUTx frequency is 180 MHz (5.55 ns period).
- The FPGA DspEclkOut pad to input flip-flop routing delay is ~2.430 ns (A).
- The FPGA DspEclkOut pad to RAMB16 routing delay is ~2.600 ns (B).
- The difference between B and A is 0.170 ns.
- The worst-case delay between the flip-flops and RAMB16 is ~2.650 ns.
  0.513 ns (Tickq) + 1.830 ns (net) + 0.311 ns (Trcck_ADDRA)
- The available time is ½ EclkOut period + 0.170 ns = ~2.95 ns.

Thus there is sufficient time (2.95 ns – 2.65 ns = 0.3 ns) to clock the flip-flops on one clock edge while clocking the block RAM on the other clock edge.

If this timing is not sufficient, one solution is to set the BUFR BUFFER_DIVIDE attribute from "BYPASS" to "1". Now the BUFR element is invoked in a divide-by-1 mode, and the timing increases from 0.212 ns to 0.732 ns. The RAMB16 components then are clocked with a skew (to the input flip-flops) of ~0.700 ns instead of ~0.200 ns.

Figure 3-8 shows the EMIF write timing waveforms for this example.

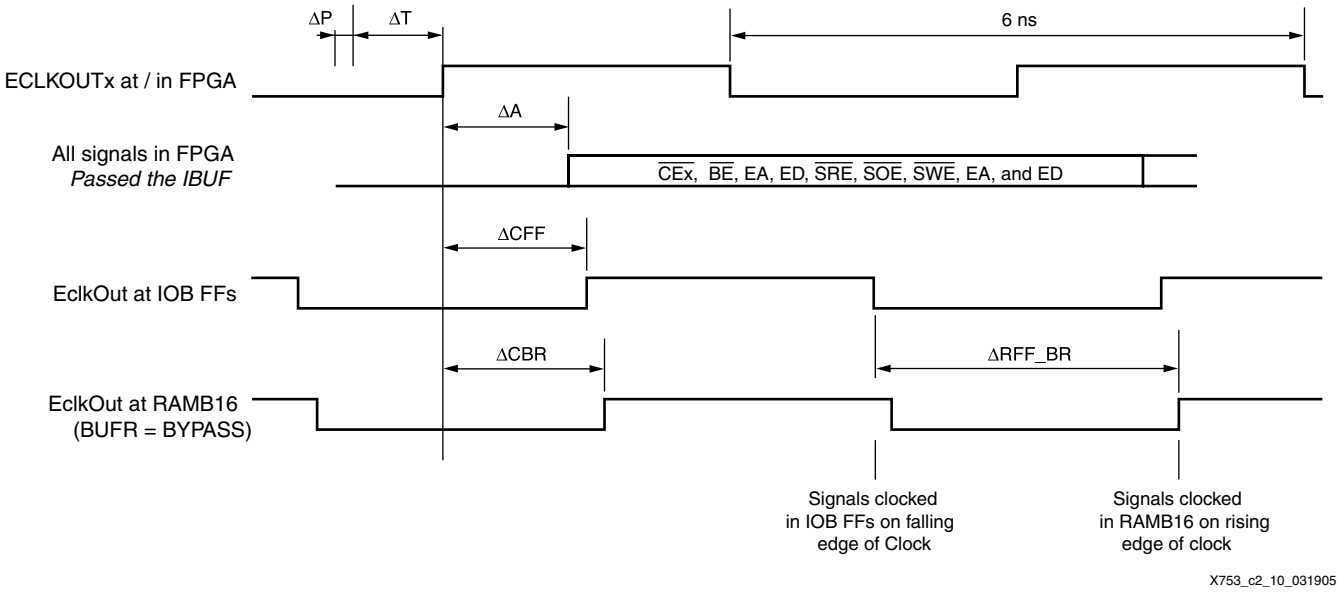


Figure 3-8: **EMIF Write Timing Diagram**

Table 3-2 defines the timing parameters shown in Figure 3-8.

Table 3-2: **Timing Parameter Descriptions for Example 2**

Parameter	Description
DSP	
ΔA	Minimum CLKOUTx to signal valid delay of the DSP (1.3 ns).
PCB	
ΔP	PCB trace delay between the DSP and the FPGA. This signal is the travel time between the pins of both devices. A trace length of 100 mm requires ~600 ps. For an EMIF read operation, this delay must be counted twice – once to address the FPGA and once to collect the data from the FPGA.
FPGA Inputs(1)	
ΔT	FPGA input delay (Tiopi = 0.954 ns). This delay is the same for all FPGA inputs. It is the time between the package pad and the output of the IOB input buffer (IBUF.O).

Table 3-2: **Timing Parameter Descriptions for Example 2 *(Continued)***

Parameter	Description
ΔCFF	Internal FPGA delay of the clock from the IBUF.O to the clock pin of the IOB flip-flops. This delay is built as: net + Tbufiocko_O + net ~0.400 ns + 0.0 ns + ~1.15 ns = ~1.5 ns
ΔCBR	Internal FPGA delay of the clock from the IBUF.O to the clock pin of the RAMB16. net + Tbufiocko_O + net + Tbrcko_O_BYP + net ~0.400 ns + 0.0 ns + 0.101 ns + 0.212 ns + ~0.990 ns = ~1.6 ns
ΔRFF_BR	Delay between the flip-flop clock edge and the RAMB16 clock edge. This delay must be greater than the routing delay from the flip-flops to the RAMB16. 3 ns > Tickoq + net + Trcck_XXX (average 2.7 ns)

Notes:

1. To calculate the delay between the DSP to the on-die IOB, add Tiopi = 0.954 ns (the value shown as ΔT in the timing diagrams) plus the package flight delay to the PCB delay.

If normal CLB (SLICE) flip-flops are used to register the EMIF clock signals, all action is taken on the regional clock distributed by the BUFR clock buffer. Then the signals are best clocked into the flip-flops on the clock's rising edge and into the block RAM on the clock's falling edge.

The EMIF read section of this interface is the same as for the interface described in Example 1. It does not use any logic between the memory and the FPGA output pads, making the interface as timing efficient as possible.

The interface has zero latency for writes and two clock cycle latency for reads.

To achieve the timing and performance listed in Figure 3-8 and Table 3-2, the CESEC and CECTL registers are configured as follows:

- CESEC register:
  - SYNCRL is set to 2 (read latency).
  - SYNCWL is set to 0 (write latency).
  - RENEN is set to 1 ($\overline{SADS}$/$\overline{SRE}$ act as $\overline{SRE}$).

    When RENEN = 1, $\overline{SRE}$ and $\overline{SWE}$ are opposites, and therefore only one of the two signals can be used to access the Virtex-4 block RAM. In this case, the $\overline{SWE}$ signal is used.
  - CEEXT is set to 1 ($\overline{CE}$ is active when $\overline{SOE}$ is active).
- CECTL register:
  - MTYPE is set to 0xE (64-bit synchronous interface for EMIFA).
  - MTYPE is set to 0x4 (32-bit synchronous interface for EMIFA).
  - MTYPE is set to 0xB (16-bit synchronous interface for EMIFA/EMIFB).

Block RAM Used as FIFOs (Example 3)

Figure 3-9 shows an example configuration where the Virtex-4 block RAM is used as a FIFO. The interface to the block RAM as a FIFO is similar to using a RAMB16 as memory.

The RAMB16 interface uses eight block RAMs, each configured as 4 bit by 4096. One port of the block RAM can be used to provide read and write access for the EMIF while the other port is used as an interface to the FPGA logic. The Virtex-4 FIFO interface differs at this point from the RAMB16 interface. It still uses eight block RAMs but is configured as 8 bits by 2048. Four FIFOs are write FIFOs, and the other four are read FIFOs.

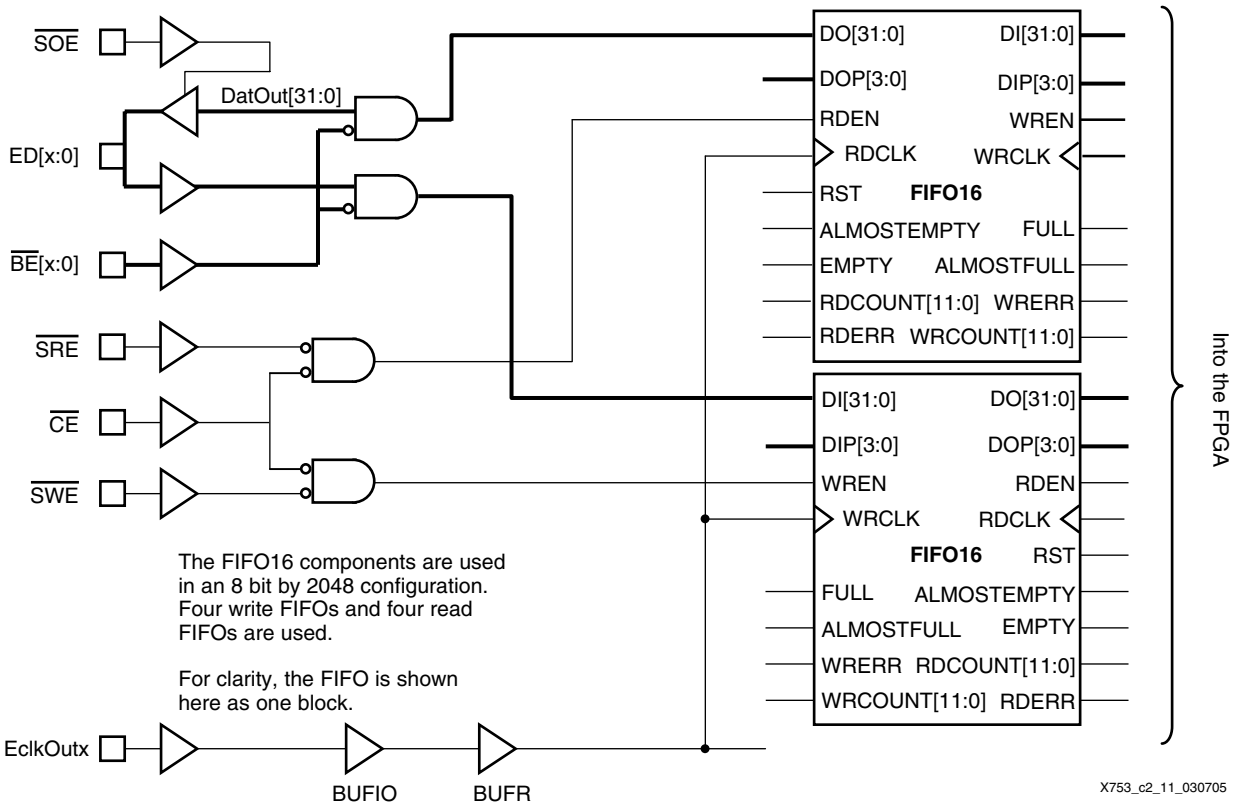


Figure 3-9: **FPGA FIFO Interface**

Because read and write blocks are separate units in the design, both $\overline{SWE}$ and $\overline{SRE}$ control the FIFO.

Figure 3-10 shows a FIFO configuration in the Virtex-4 FPGA. The FIFO interface needs fewer I/O pins than the memory interface, but the DSP still outputs address and byte enable information, which can be used, if needed.

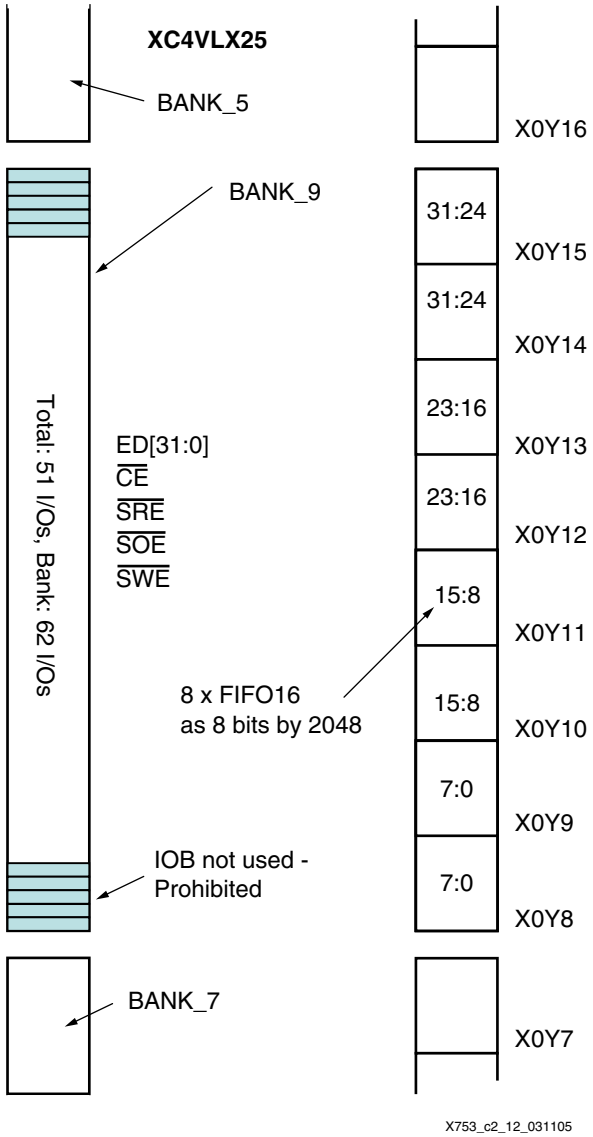


Figure 3-10: **Setup of the FIFO16 in the Virtex-4 FPGA**

The flags indicate the FIFO status to the DSP. The simplified design connects the empty (almost_empty) and full (almost_full) flags directly to the DSP interrupt pins. Another design can use an interrupt controller in the FPGA with one interrupt connected to the DSP.

Figure 3-11 and Figure 3-12 show the EMIF write and read timing waveforms for this example, respectively.

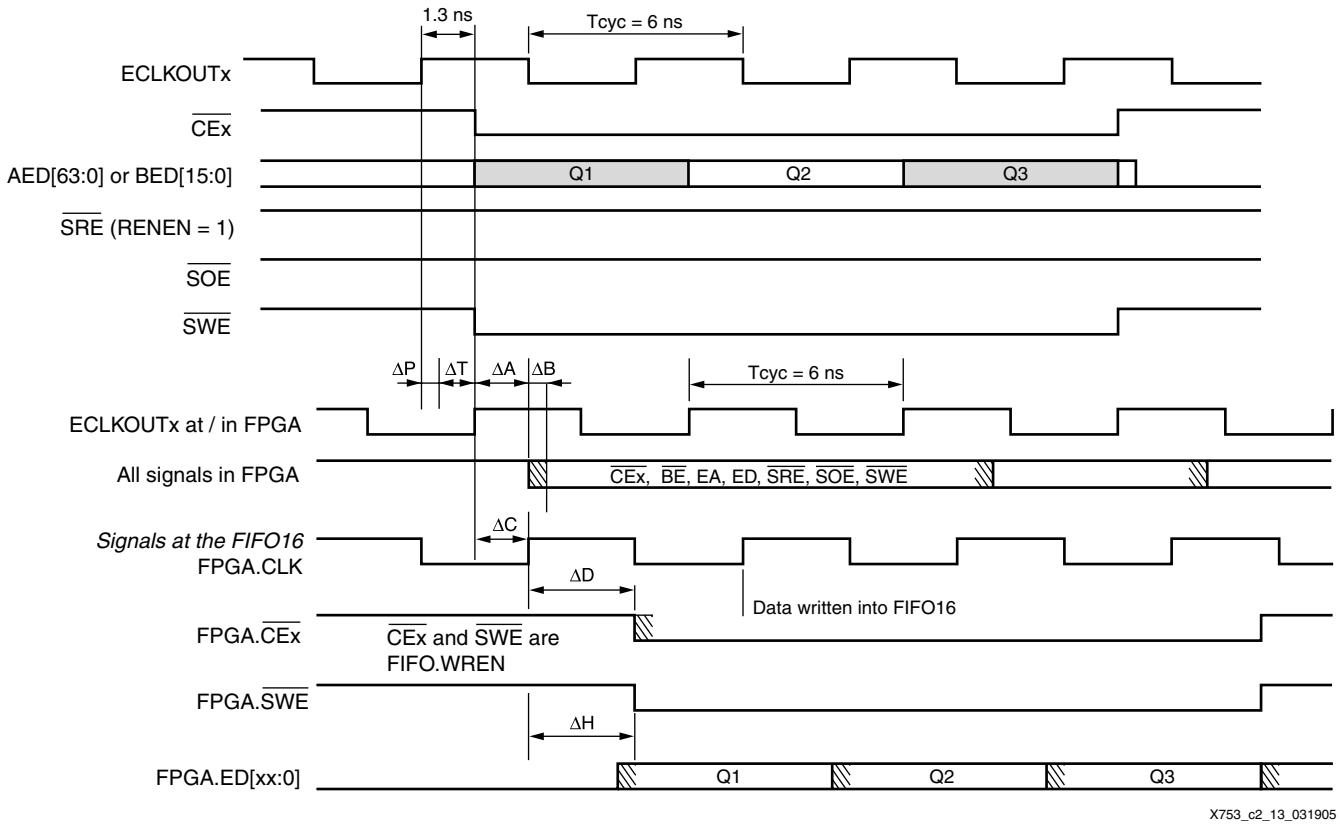


Figure 3-11: **FIFO Interface Write Timing Diagram**

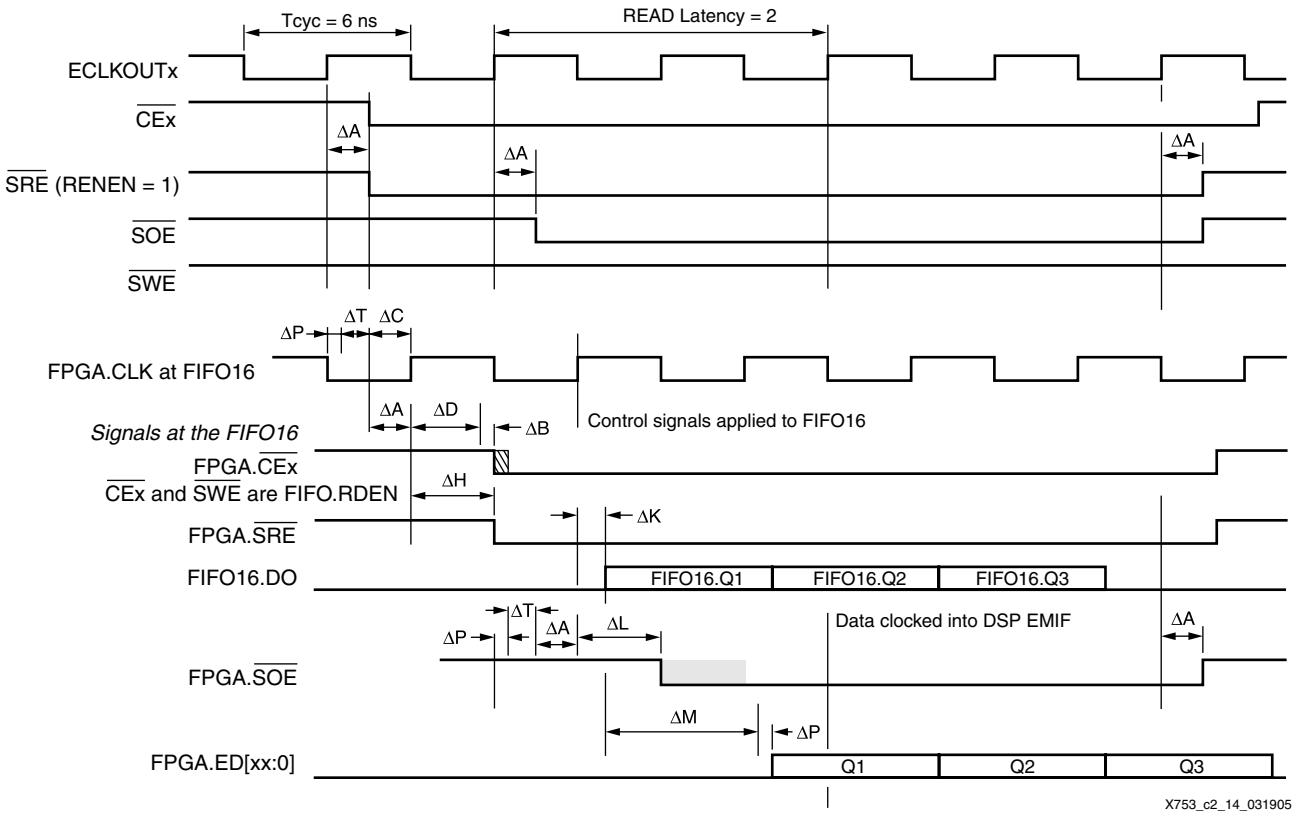


Figure 3-12: **FIFO Interface Read Timing Diagram**

Table 3-3 defines the timing parameters shown in Figure 3-11 and Figure 3-12. The timing parameters are slightly different for the FIFO16 interface than for the RAMB16 interface.

Table 3-3: **Timing Parameter Descriptions for Example 3**

Parameter	Description
FPGA Inputs(1)	
ΔD	Internal FPGA delay from the IBUF.O of the $\overline{CE}$ pin to the RAMB16. net + Tilo + net + Trcck_ENA ~1.5 ns + 0.194 ns + ~0.9 ns + 0.403 ns = ~3 ns
ΔE	Internal FPGA delay from the IBUF.O of the $\overline{BE}$ pins to the RAMB16. net + Tilo + net + Trdck_DIA ~1.2 ns + 0.194 ns + ~0.8 ns + 0.115 ns = ~2.3 ns
ΔF	IBUF.O of ED data pins to the RAMB16 memory. net + Tilo + net + Trdck_DIA ~1.1 ns + 0.194 ns + ~0.9 ns + 0.115 ns = ~2.3 ns
ΔH	Internal FPGA routing and setup for WEA ($\overline{SWE}$ and $\overline{SRE}$) signal. net + Tilo + net + Trcck_WEA ~1.1 ns + 0.194 ns + 0.9 ns + 0.630 ns = ~2.8 ns

Table 3-3: **Timing Parameter Descriptions for Example 3 *(Continued)***

Parameter	Description
FPGA Output	
ΔM	Data from RAMB16.O outputs to output package pad. net + Tilo + net + Tioop ~1.3 ns + 0.194 ns + 0.9 ns + 2.961 ns = ~5.4 ns

Peripheral Device Transfer (PDT) Interface (Example 4)

In this example, the FPGA is not directly used for data with the DSP. Instead it is mainly used to transfer data to and from external memory connected to and used by the DSP. In this case, the EMIF can play the role of DMA engine.

Usually too many DSP operations are used in this situation. When transferring data from the FPGA to memory, the process is:

1. The EMIF reads the FPGA.
2. EMIF stores data internally.
3. EMIF writes to the memory.

This sequence is repeated as long as there is data to transfer between the two devices. When the EMIF plays the role of DMA engine, the data flows directly from the FPGA to memory or from memory to the FPGA.

The PDT interface is only supported when the external memory is SDRAM and the peripheral (FPGA) is not memory mapped. A PDT transaction has the following characteristics:

- The EMIF generates normal SDRAM read and write bus cycles.
- The EMIF generates a $\overline{\text{PDT}}$ signal, based on the settings of the PDTWL and PDTRL fields in the PDTCTL register.
- The two MST address lines are used as extra control lines during a PDT transaction:
  - $\overline{\text{PDTA}}$ (PDT access)

    Depending on the bus width, $\overline{\text{PDTA}}$ is EA19 (64 bits), EA18 (32 bits), or EA17 (16 bits).
  - PDTD (PDT direction)

    Depending on the bus width, PDTD is EA20 (64 bits), EA19 (32 bits), or EA18 (16 bits).
- The EMIF data lines ED[x:0] are held in a high-impedance state, allowing other devices to take control of the bus.

The simplest solution is to use the FPGA interface with FIFO16s as shown in Figure 3-13. Figure 3-14 shows the FPGA logic block in Figure 3-13 in more detail.

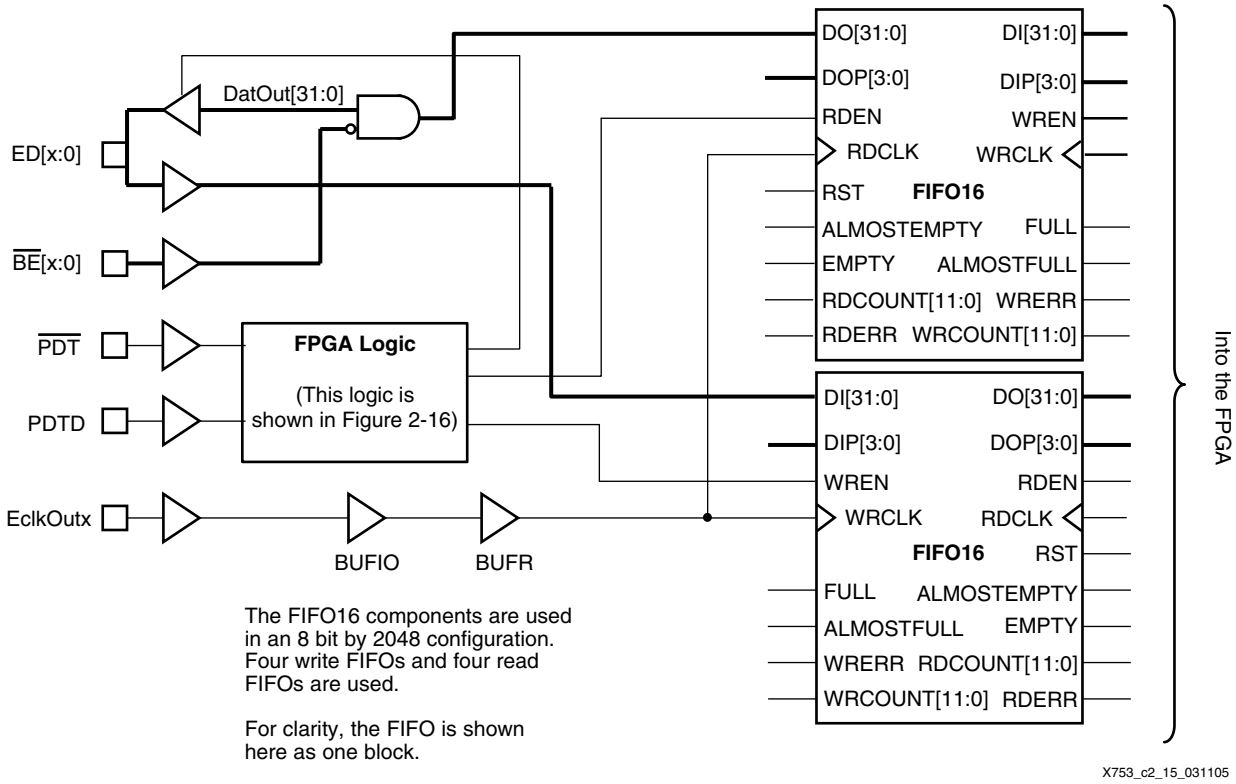


Figure 3-13: **EMIF FPGA Connection for PDT Transactions (Example 4)**

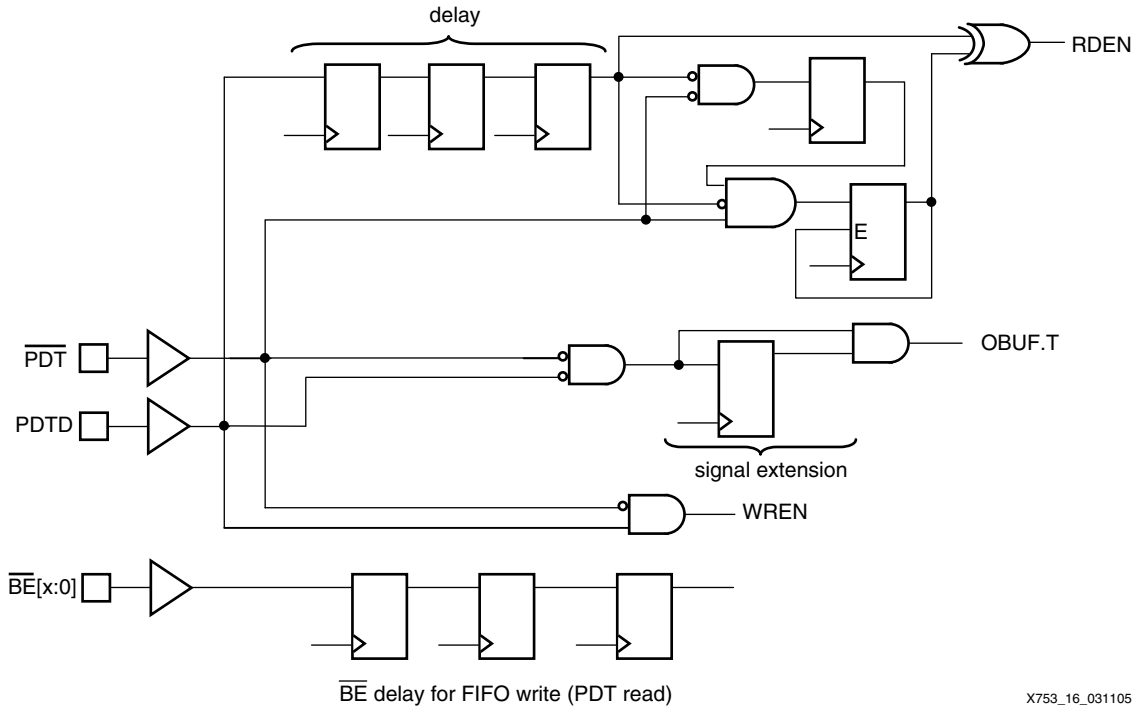


Figure 3-14: **PDT Logic in the FPGA**

The Virtex-4 FPGA logic allows the PDT interface to be used in place of a normal FIFO EMIF. For the PDT read interface (that is, FIFO16 write interface), the PDTRL setting in the PDTCTL register is set to 0 (zero latency). This way the data can be written directly into the FIFO16 when $\overline{\text{PDT}}$ is Low and PDTD is High.

The BE bits are delayed by three ECLKOUTx clock cycles through the consecutive flip-flops (as shown in Figure 3-14).

The PDT write interface needs some extra logic, too. To simulate a two-latency FIFO16 read, the PDTD signal must be delayed for three clock cycles.

The PDTWL bit in the PDTCTL register must be set to 1 to properly configure the $\overline{\text{PDT}}$ signal, which needs an extension of one clock cycle. Some logic also must be constructed to let PDTD rise when $\overline{\text{PDT}}$ is rising.

This timing occurs when the FIFOs used are the closest to the FPGA I/O. When FIFO16s or RAMB16s in the middle of the FPGA are used, signals need more or less delay, or an extra extension, which can easily be achieved with additional delay flip-flops.

The timing of this setup occurs under complete control of the DSP EMIF registers. The use of SDRAM requires the setting of the SDCTL, SDTIM, and SDEXT registers, while having the PDT requires the setting of the PDTCTL and EDMA registers.

Figure 3-15 shows the waveforms for a PDT write with the configuration shown in Figure 3-13. The data is transferred from the FPGA FIFO to an external SDRAM memory under control of the DSP EMIF.

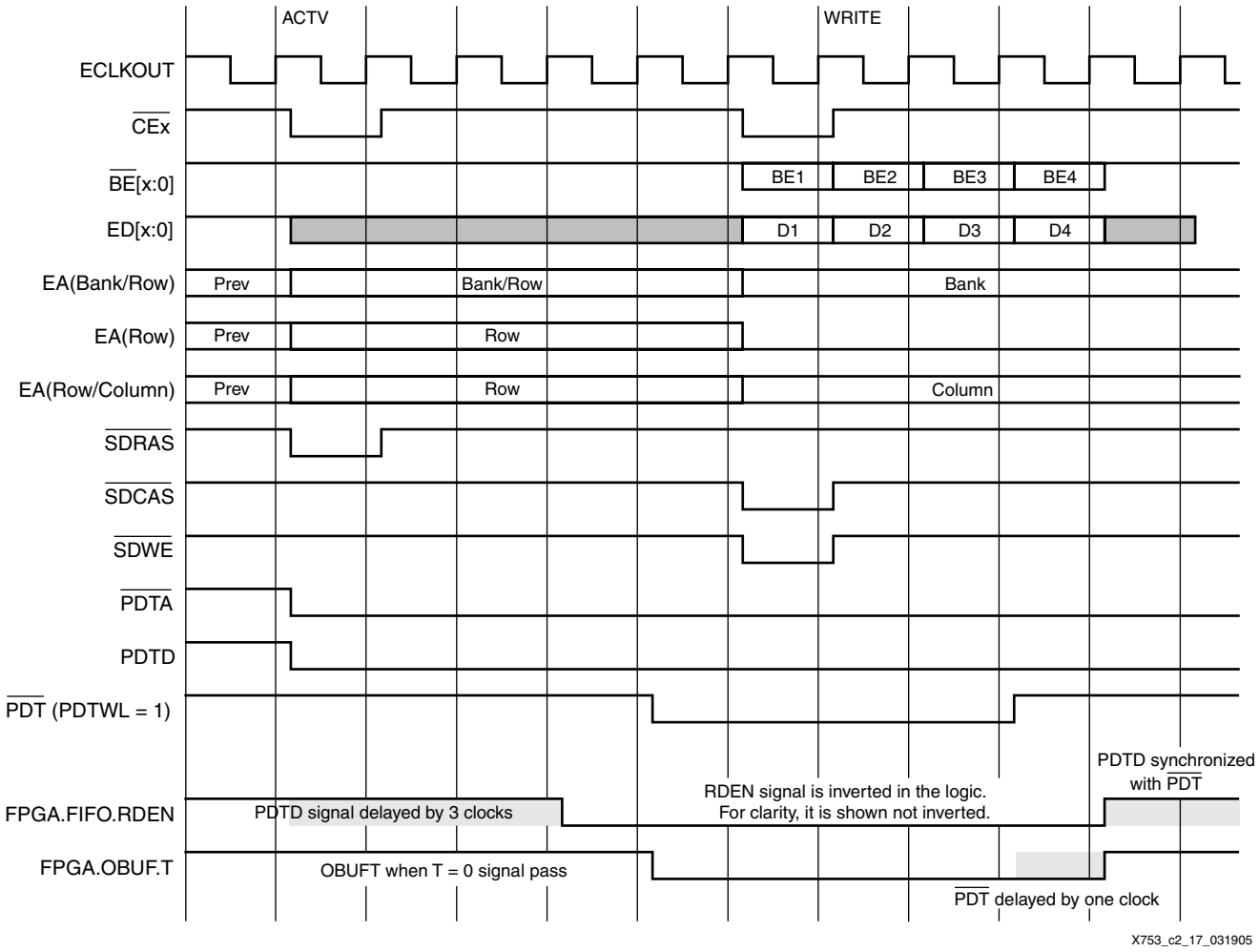


Figure 3-15: **PDT Write Timing**

A PDT write transfer has the following characteristics:

- For EMIF access to the SDRAM, the $\overline{\text{CEx}}$ of the addressed space is active.
- The PDT drives the PDTD and $\overline{\text{PDTA}}$ control signals during the ACTV cycle. This action happens at the same time that the address appears on the bus.
  - PDTD is driven Low, indicating a FIFO read because the data transfer happens from FIFO to memory.
  - $\overline{\text{PDTA}}$ is driven Low.
- Normal write control signals are generated for the SDRAM except:
  - $\overline{\text{PDT}}$ is driven Low under control of the PDTWL settings in the PDTCTL register.
  - The EMIF data pins are high-impedance, and therefore the SDRAM captures the data from the FIFO.

Figure 3-16 shows the waveforms for a PDT read with the configuration shown in Figure 3-13. The data is transferred from the external SDRAM into the FPGA FIFO under control of the EMIF.

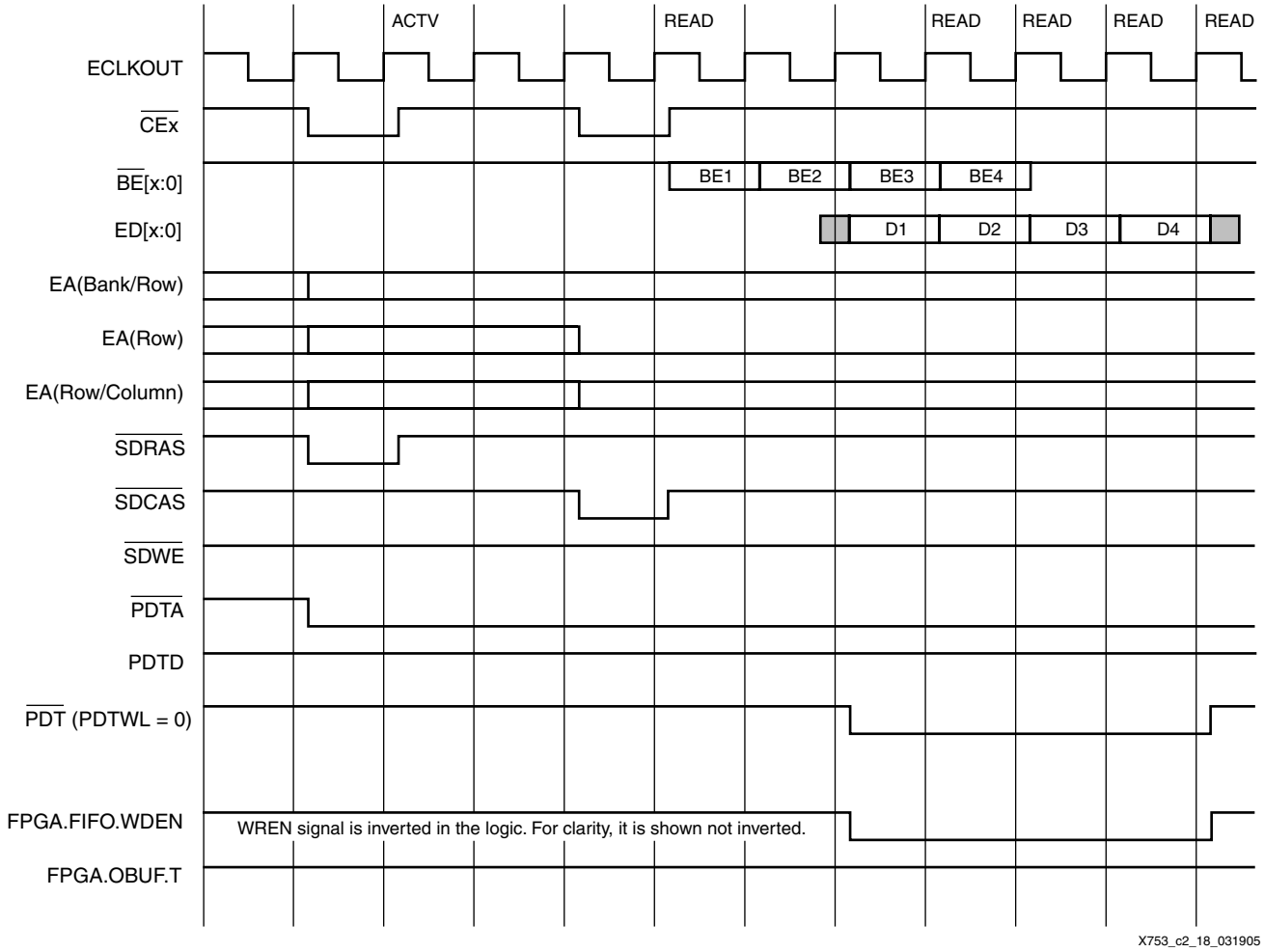


Figure 3-16: **PDT Read Timing**

A PDT read transfer has the following characteristics:

- For EMIF access to the SDRAM, the $\overline{CEx}$ of the addressed space is active.
- The PDT drives the $\overline{PDTA}$ control signals during the ACTV cycle. This action occurs at the same time that the address appears on the bus.
  - PDTD remains High, indicating a FIFO write because the data transfer comes from memory to FIFO.
  - $\overline{PDTA}$ is driven Low.
- Normal read control signals are generated for the SDRAM except:
  - $\overline{PDT}$ is driven Low under control of the PDTRL settings in the PDTCTL register.
  - EMIF data is ignored.

It is possible to combine a memory type interface and a PDT FIFO type interface in the FPGA through the same EMIF I/O connections as shown in Figure 3-17.

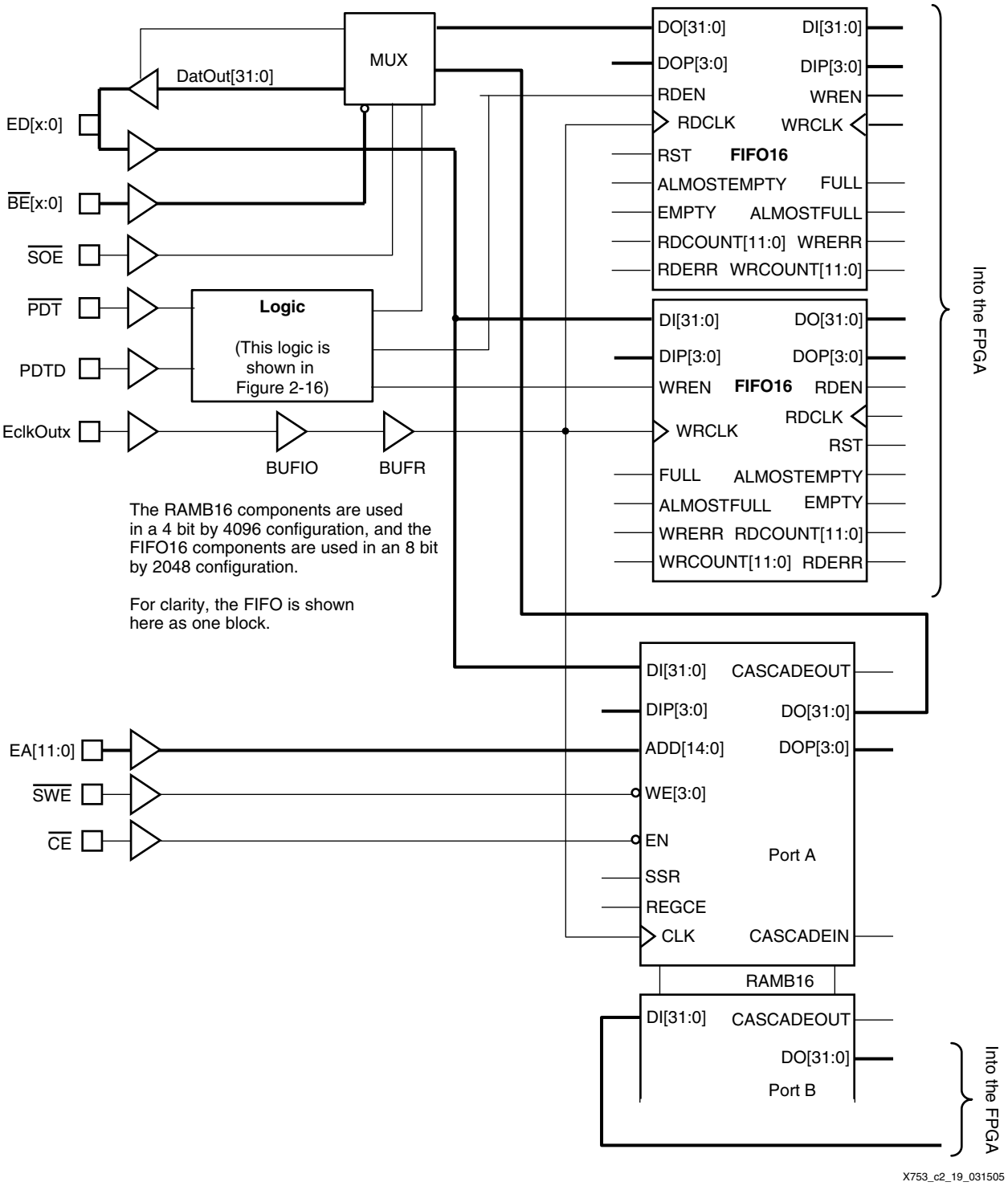


Figure 3-17: **Memory and $\overline{\text{PDT}}$ FIFO**

In this combination, the FPGA and the EMIF port can communicate (read and write) with the RAMB16 memory, and the FIFO16 operates under PDT operation.

Because PDT and normal memory work with different control signals, both can be accessed in the same memory space.

The memory is smaller than the addressable EMIF memory space. The MSB address lines can be kept Low during normal memory access, and function as control lines during PDT transfers. With the PDT control signal $\overline{\text{PDTA}}$, the normal memory control lines $\overline{\text{SWE}}$ and $\overline{\text{SRE}}$ can be blocked.

Switching from an EMIF memory access type interface to a PDT type interface requires that the EMIF registers of the DSP be changed accordingly.

Virtex-4 IOB with ISERDES and OSERDES Functionalities

Virtex-4 ISERDES

The Virtex-4 ISERDES contains all possible functions of the input block, including the IDELAY component. The ISERDES can be a serializing input register with parallel latch registers into the FPGA logic. The serializer function has a length of one bit to six bits, or, when two ISERDES elements are used in master-slave mode, the length is 10 bits.

The serializer register is clocked at *x* times the clock rate of the parallel latching register, where *x* is the number of serial shifted bits.

The ILOGIC component is a subset of the ISERDES element. ILOGIC is the combination of IDELAY and IDDR flip-flops. ILOGIC comprises the first two registers of the SERDES device (in DDR mode, the first four registers). The ILOGIC block must be used to obtain direct flip-flop outputs without the parallel latch register. Figure 3-18 through Figure 3-20 show graphical representations of both ILOGIC and ISERDES functionalities in different modes. Because the EMIF is a single-ended signaling interface, only SDR modes are shown.

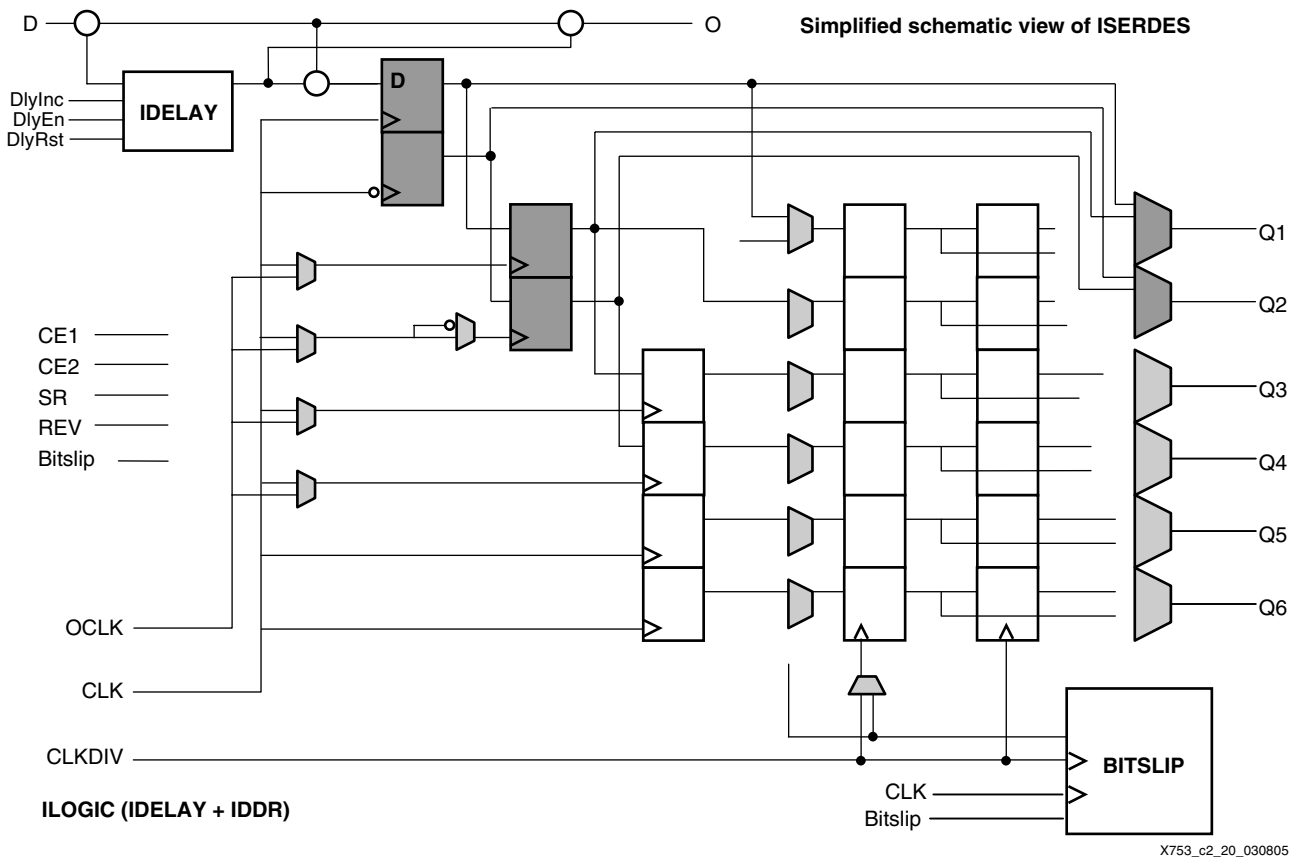


Figure 3-18: **ILOGIC (ISERDES and IDDR)**

Table 3-4 lists the ILOGIC attributes and values.

Table 3-4: **ILOGIC Attributes**

Attribute	Possible Values
DDR_CLK_EDGE	OPPOSITE_EDGE, SAME_EDGE, SAME_EDGE_PIPELINED
INIT_Q1	0, 1
INIT_Q2	0, 1
SRTYPE	ASYNC, SYNC

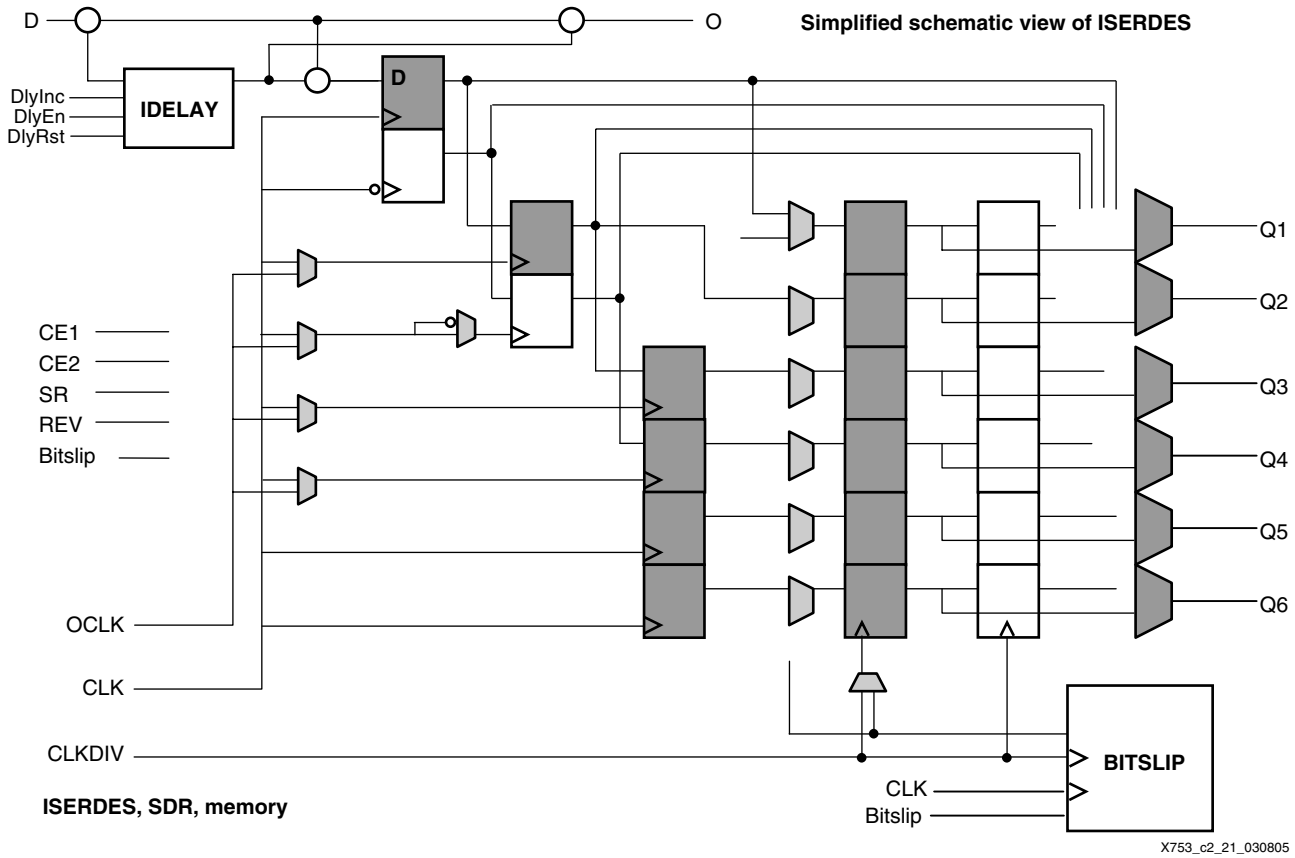


Figure 3-19: **ISERDES: SDR Memory Mode**

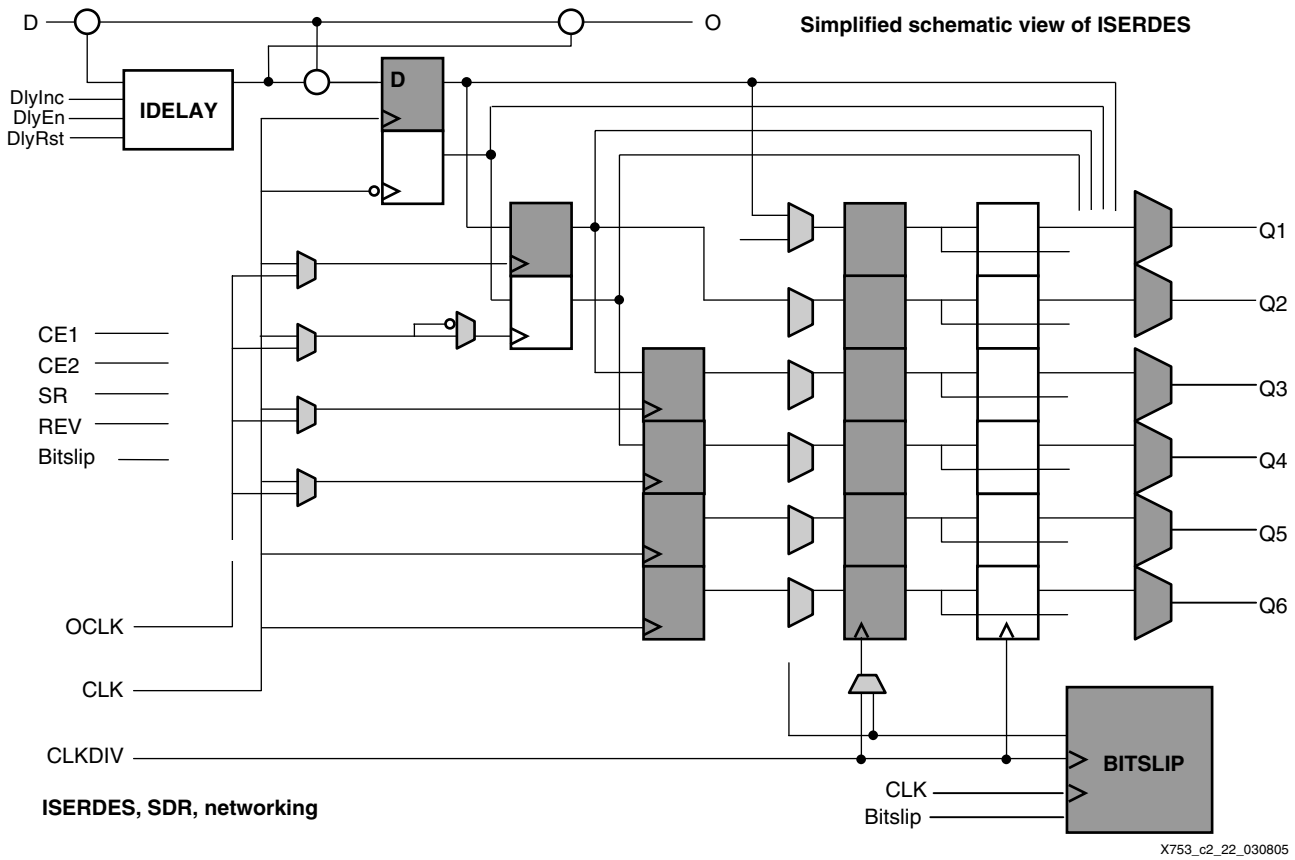


Figure 3-20: **ISERDES: SDR Networking Mode**

Table 3-5 lists the ISERDES attributes and values.

Table 3-5: **ISERDES Attributes**

Attribute	Possible Values
BITSLIP_ENABLE	TRUE, FALSE
DATE_RATE	SDR, DDR
DATA_WIDTH	2, 3, 4, 5, 6, 7, 8, or 10
INTERFACE_TYPE	MEMORY, NETWORKING
IOBDELAY	NONE, IBUF, IFD, or BOTH
IOBDELAY_TYPE	DEFAULT, FIXED, or VARIABLE
IOBDELAY_VALUE	0 to 63
NUM_CE	1 or 2
SERDES_MODE	MASTER or SLAVE
INIT_Q1	0, 1
INIT_Q2	0, 1
SRTYPE	ASYNC, SYNC

Virtex-4 OSERDES

This implementation uses the OSERDES functionality shown in Figure 3-21 and Figure 3-22. It does not support direct output, master and slave configuration, and DDR flow through the OSERDES. For displaying purposes, only the data side of the output block is shown. A 3-state function similar to the data output functionality exists for every single I/O.

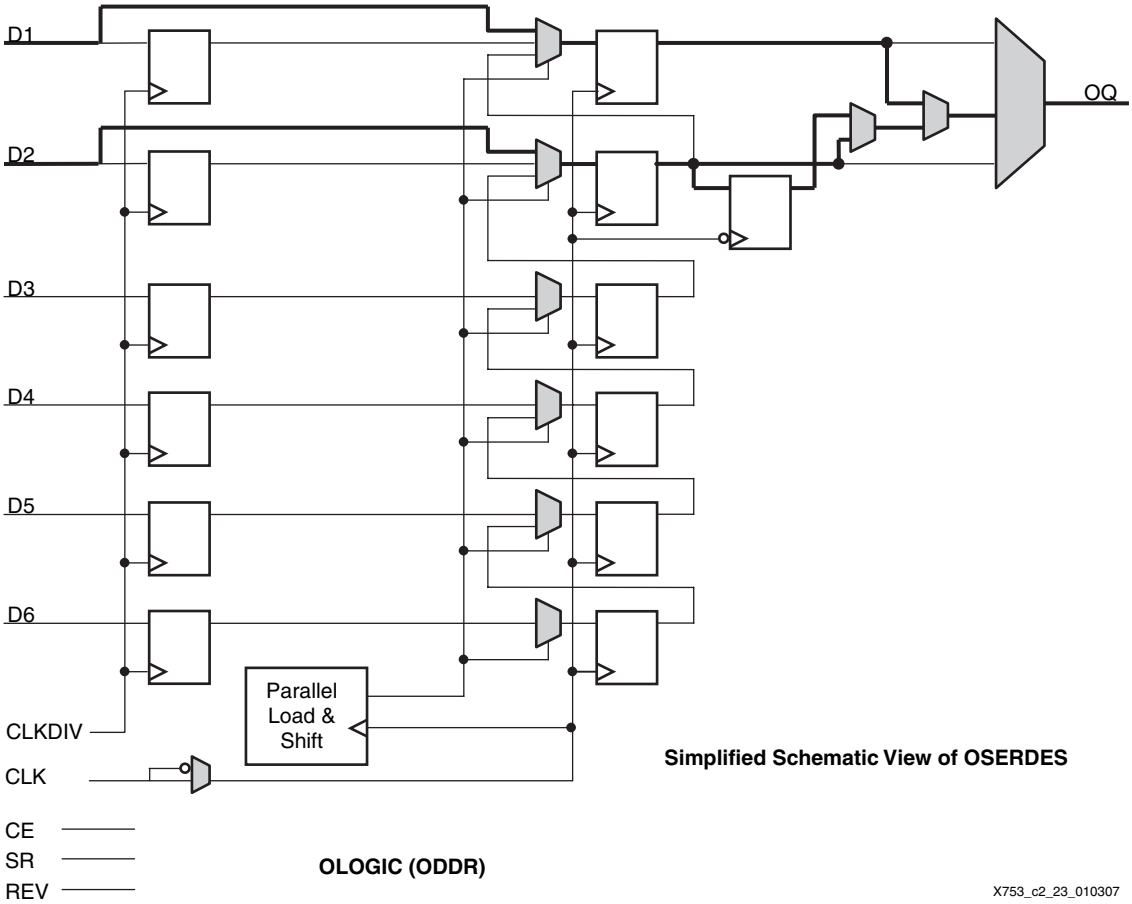


Figure 3-21: **OLOGIC (ODDR)**

Table 3-6 lists the OLOGIC attributes and values.

Table 3-6: **OLOGIC Attributes**

Attribute	Possible Values
DDR_CLK_EDGE	OPPOSITE_EDGE, SAME_EDGE
INIT	0, 1
SRTYPE	ASYNC, SYNC

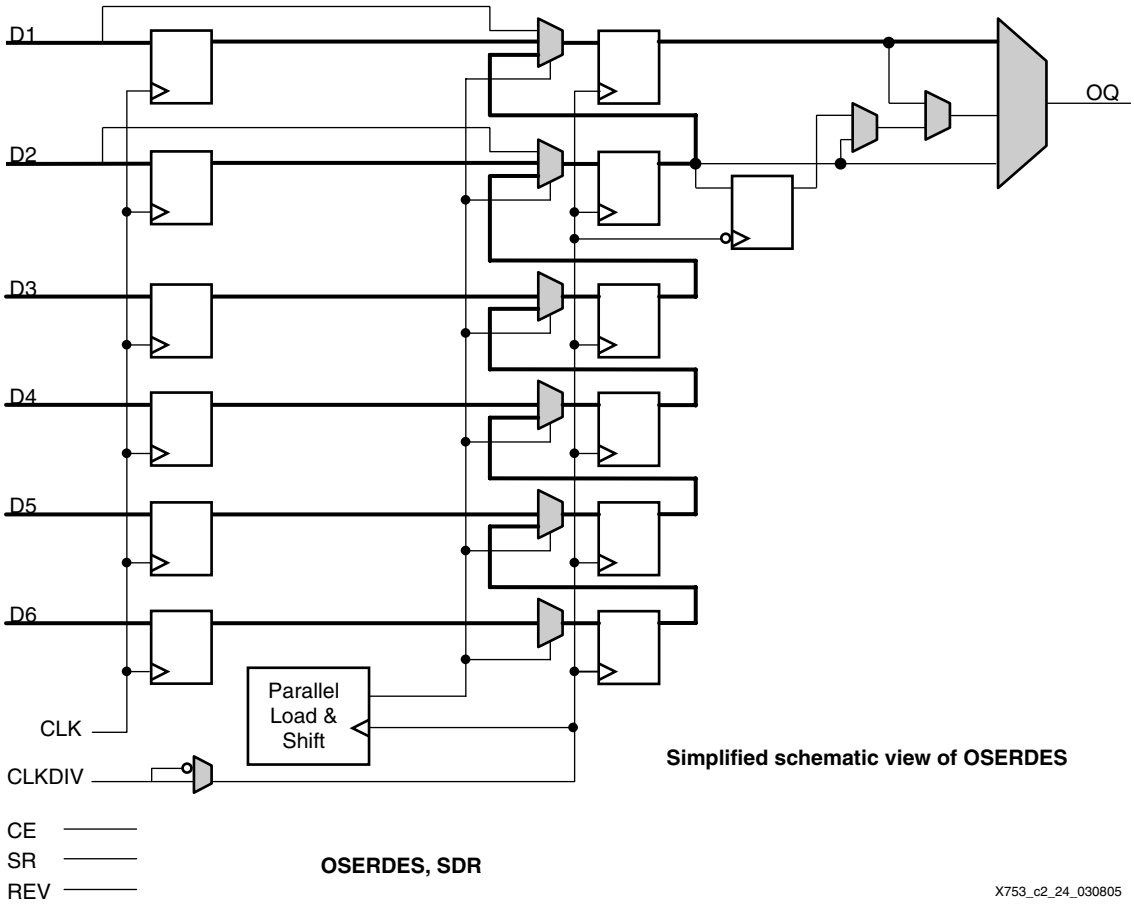


Figure 3-22: **OSERDES**

Table 3-7 lists the OSERDES attributes and values.

Table 3-7: **OSERDES Attributes**

Attribute	Possible Values
DATA_RATE_OQ	SDR, DDR
DATA_RATE_TQ	BUF, SDR, or DDR
DATA_WIDTH	2, 3, 4, 5, 6, 7, 8, or 10
SERDES_MODE	MASTER or SLAVE
TRISTATE_WIDTH	1, 2, 4

Data Burst Interface

Because the I/O of the FPGA contains a serial-to-parallel and parallel-to-serial converter, this interface can be built as a data-burst-capable FPGA to EMIF. Some simple logic, a counter, and several gates are needed to build it.

This interface, also called “write and forget”, allows data to be written at a contiguous rate or burst rate. The data is written to the ISERDES in <DATA_WIDTH> shift mode, where DATA_WIDTH references the DATA_WIDTH attribute of the ISERDES and OSERDES

blocks. When data, address, or control signals are written into the ISERDES, a counter generates the load pulse and terminal count to store the value from the shift register into the parallel register. The opposite edge of the ISERDES clock is then used with the counter's terminal count to trigger the load of a shift register built in CLB flip-flops. Next, clock-edge data is shifted from this internal shift register to RAMB16 or FIFO16 inputs. New data is shifted into the ISERDES shift register on the next rising clock edge.

This way a circular data transport is organized. Serial data is input from external to the FPGA, converted to parallel data, and then serialized internally. Figure 3-23 (block diagram) and Figure 3-24 (implemented setup) show a four-bit setup of the design.

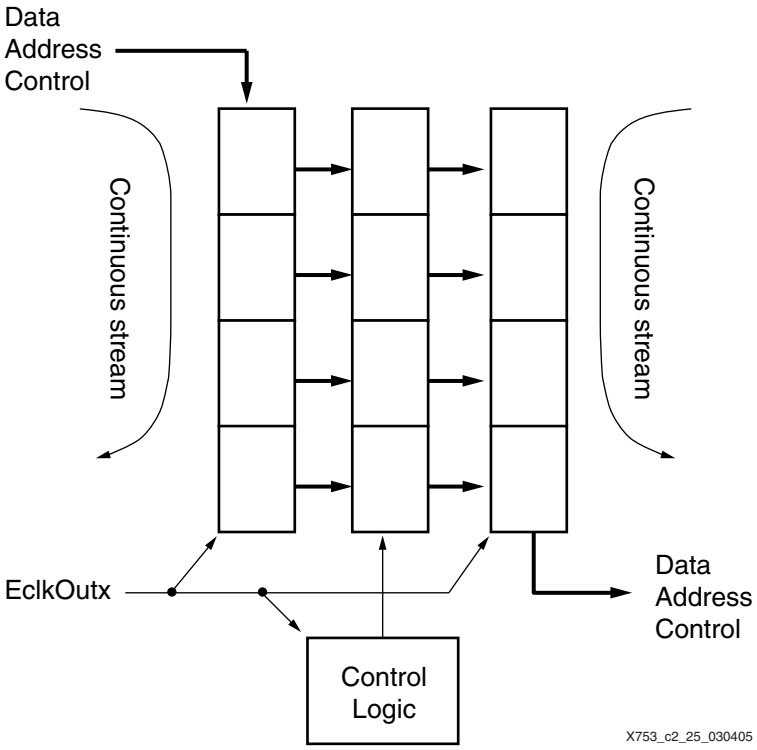


Figure 3-23: **Burst Data Transport**

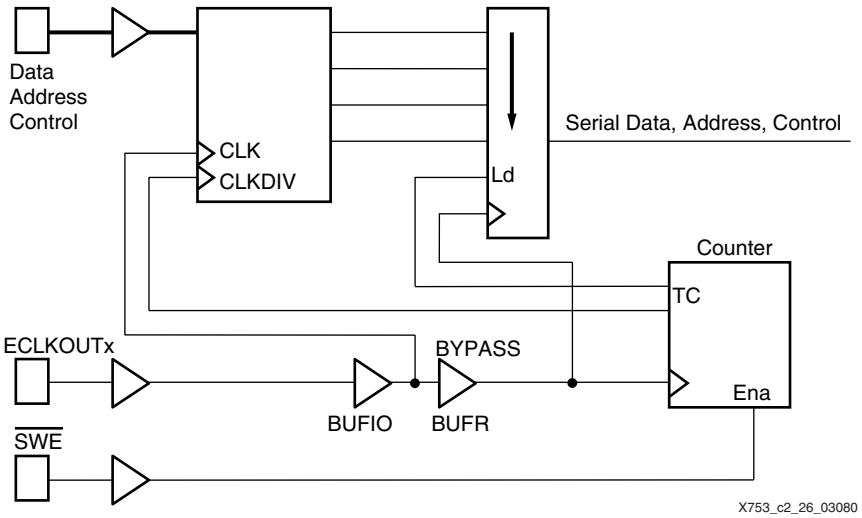


Figure 3-24: **ISERDES EMIF Design Setup**

All signals of the EMIF are input into the serial-to-parallel-to-serial register. This way the external EMIF is completely disconnected from the internal interface of the FPGA.

The signals that control the interface are split in the ISERDES by the registers and the direct logic input.

The IDELAY of the ISERDES component is not used in this chapter, but can be used to level out possible PCB trace length differences.

Because the EMIF is not a synchronous interface providing continuous data and framing signals, the best way to use the IDELAY logic is with the FIXED timing delay. TDR measures the PCB trace length differences for all EMIF I/O connections and fills the appropriate numbers into the ISERDES attribute list (see Figure 3-25). An example how this can be done is illustrated in Appendix A, "Virtex-4 ISERDES Sample Code."

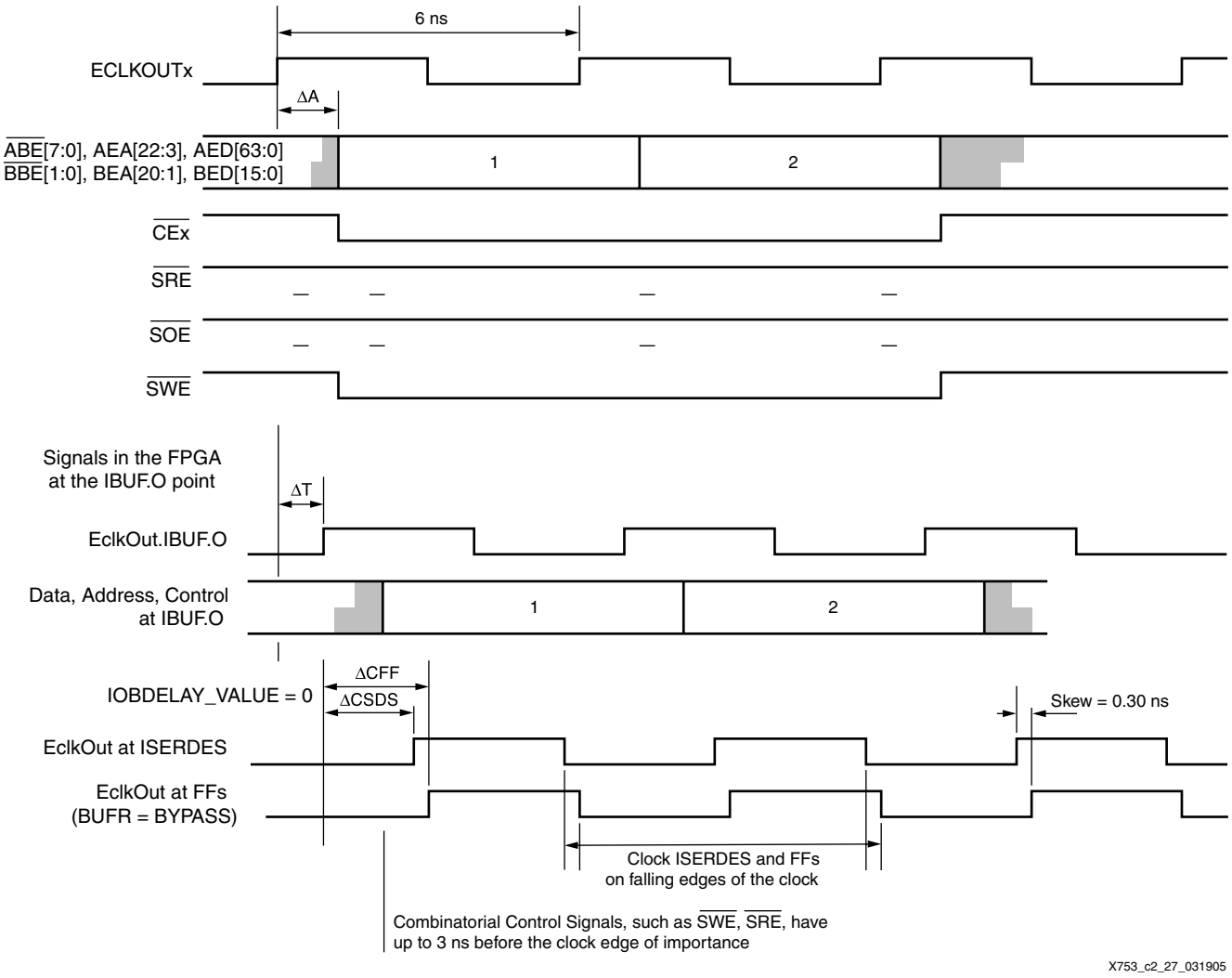


Figure 3-25: **Interface Timing Diagram**

Table 3-8 defines the timing parameters shown in Figure 3-25.

Table 3-8: **Timing Parameter Descriptions**

Parameter	Description
DSP	
ΔA	Minimum CLKOUTx to signal valid delay of the DSP (1.3 ns).
FPGA Inputs(1)	
ΔT	FPGA input delay (Tiopi = 0.954 ns). This delay is the same for all FPGA inputs. It is the time between the package pad and the output of the IOB input buffer (IBUF.O).
ΔCFF	Internal FPGA delay of the clock from the IBUF.O to the clock pin of the IOB flip-flops. This delay is built as: Tidid + Tbufiocko_O + net + Tbrcko_O_BYP + net_to_FFs = ~2.4 ns

Table 3-8: **Timing Parameter Descriptions *(Continued)***

Parameter	Description
ΔCSDS	Internal FPGA delay of the clock from the IBUF.O to the clock pin of the ISERDES. This delay is built as: Tidid + Tbufiocko_O + net + net_to_serdes = ~2.1 ns

Notes:

1. To calculate the delay between the DSP to the on-die IOB, add Tiopi = 0.954 ns (the value shown as ΔT in the timing diagrams) plus the package flight delay to the PCB delay.

Analyses of the waveforms in Figure 3-25 show the following:

- The rising clock edge for the ISERDES arrives at the same time that the input data becomes valid.
- The EMIF ECLKOUTx to data valid delay is 1.3 ns minimum.
- In the FPGA, all signals are delayed through the input buffer (Tiopi = 0.954 ns).
- The data arrives at the ISERDES.D input at 2.25 ns.
- The Eclkout clock in the FPGA passes through the BUFIO clock buffer for the ISERDES devices. It gets a delay of 2.271 ns from the pad of the ISERDES.C input.

Because this timing is too close to create a saved design, the decision is to clock the ISERDES and flip-flops of the interface on the falling edge of the clock and/or add an IDELAY element in the clock distribution. The clock delay then becomes:

Tiopi + Tidid + Tbufiocko_O + net to ISERDES.C

Tiopi + Tidid + Tbufiocko_O + net + Tbrcko_O_BYP + net to flip-flops

One way the DSP can do an EMIF read operation is under control of the FPGA interface, making the FPGA interface a “write and forget” operation for the FPGA back-end design logic or processor (PPC405 or MicroBlaze processor). When the OSERDES is loaded with data, an interrupt signal for the DSP is generated. The DSP interrupt routing can start an EMIF read operation and empty the OSERDES. Once the OSERDES is empty, the whole process can restart.

Register Initialization Interface

Part of the above interface can be used as a unidirectional interface to initialize registers, for processor or other use, and as a simple and fast way to communicate with the PPC405 or MicroBlaze processor.

The entry part of the design can be the same as the Data Burst interface (see “Data Burst Interface”), where data is written into the ISERDES shift register under control of the EMIF control lines. Once the shift register is full, data is transferred to other registers in the Virtex-4 FPGA application design.

This approach, where the DSP initializes registers in the FPGA, does not need address lines. All data for the registers can be written for the DSP at the same address. In the FPGA logic, the registers, possibly used with the PPC405 or MicroBlaze processors, reside at different addresses.

Figure 3-26 shows a possible setup.

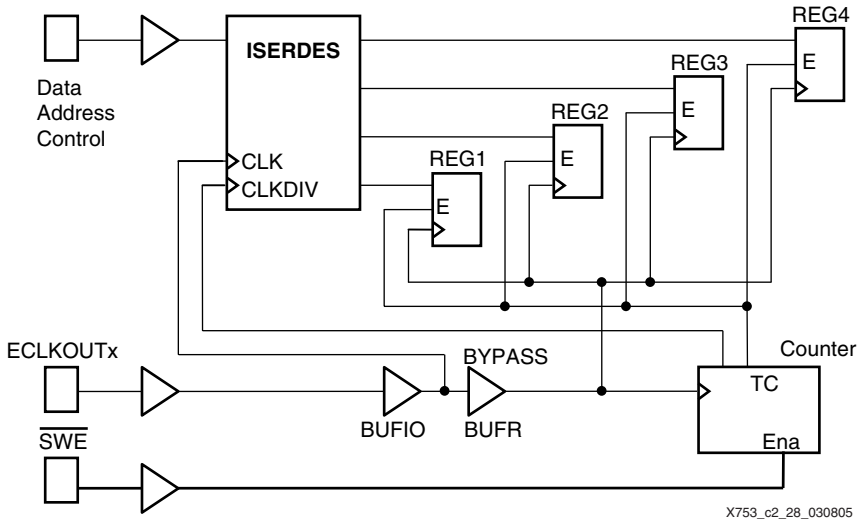


Figure 3-26: **Unidirectional Register Initialization Interface**

When data needs to be communicated from the Virtex-4 FPGA to the DSP, the same approach can be used.

Printed Circuit Board (PCB)

This section discusses PCB design issues.

Component Placement

Different circuit components must be placed as close as possible to each other and, if possible, lined up with respect to the I/O pinning. Components must be placed close by for limited PCB trace length, corners, and pass-through PCB vias.

FPGA pin assignment is very flexible, because any pin can provide the required functionality.

A straight, short connection improves all parameters of a PCB layout:

- Signal integrity
- Transmission line effects
- Capacitance and inductance
- Operating frequency

If PCB traces have different lengths, the IDELAY block in the ISERDES devices can level the differences. Measure the traces in the PCB routing software and calculate the differences in delay between the traces. Use the IDELAY in “IFD”, “FIXED” mode and provide an integer number (the delay appears in 87 ps stubs) to cancel the delay differences of the traces (see Figure 3-27).

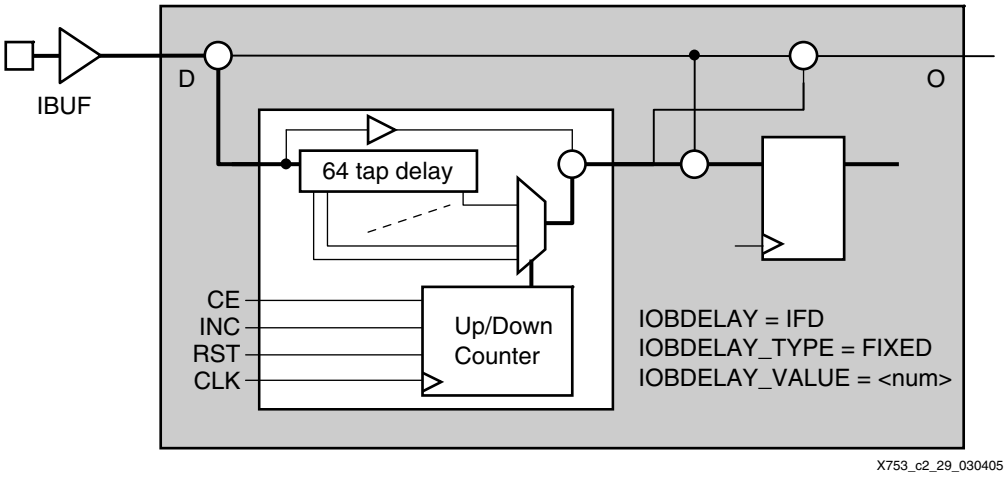


Figure 3-27: **IDELAY Initialized as IFD, FIXED**

If distances between components are long, transmission line effects play a role. All transmission lines must be terminated properly to control reflections.

Careful component placement, early in the design process, on the PCB using the layout software produces a better PCB design. Before a PCB layout is started, the number of layers to be used needs to be determined as does the destination of the layers (signal, ground, power, and so forth).

PCB Guideline Summary

The following bulleted list summarizes the guidelines for PCB design:

- Spend a sufficient amount of time when placing components for layout. Shuffle the blocks for the best design.
- Keep the trace lengths as short as possible.
- Spend time determining the number of PCB layers and how the layers stack.
- If possible during PCB layout, keep the length of a track shorter than the travel and reflection time of the signal on the trace. If not possible, use transmission line theory.
- Route traces on the PCB far from each other to avoid or minimize crosstalk effects.
- Do not route traces into 90 degree turns, and never route traces into 180 degree turns. 90 degree turns increase the effective width of the trace, contributing to parasitic capacitance. At very fast edge rates (~100 ps), these discontinuities can cause significant signal integrity problems.
- Use round, circular turns. If this is not possible, use 45 degree corners.
- Take the guidelines of *Signal Return Paths* into account.
- Guard traces.
- Remember that Ground planes are very important.
- Pay attention to power distribution. Make sure the ground and power planes are well connected with many bypass capacitors to create low-inductance signal return paths.
- Dead end stubs. Never allow a trace stub to exist without termination. This kind of stub is an antenna, and the uncontrolled impedance causes signal reflections that result in unpredictable behavior.

- PCB trace turns. Never make a trace turn 180 degrees on a PCB. This way a perfect antenna is constructed. Therefore a board should never have traces that turn more than 45 degrees.

Note: Antennas both emit and receive signals.

- When dealing with transmission lines and using freely available calculation programs, be aware that not all published formulas give the same result. Most formulas are put together empirically and then fine-tuned by others or even more experimental work.
- Take the average out of different calculations, done with different programs and tools. All this "stuff" has a high level of "Black Magic".
- As with all things, experience helps! If you have no experience at all, try to find somebody with some or take some good training courses.

Conclusion

Interfacing the Texas Instruments C64x DSP family to Virtex-4 FPGAs is an easy and straightforward process. Xilinx ISE tools are used to generate logic, I/O, and memory devices. Both the FPGA and DSP have flexible timing parameters allowing minimal logic and achieving high performance.

With minimal design efforts, the FPGA can perform a coprocessor/preprocessor, high-speed data processing, or high-speed data bridge function for the DSP.

XILINX®

Chapter 4

Reference Designs

Detailed explanations of the reference designs can be found in the `readme.txt` file, located in the respective ZIP file.

The reference design files for the Virtex-II, Spartan-3, and Virtex-4 implementations can be downloaded from the Xilinx website at:

http://www.xilinx.com/bvdocs/appnotes/xapp753.zip

XILINX®

Appendix A

Virtex-4 ISERDES Sample Code

This appendix provides sample code for engineers who are designing a Virtex-4 based system with a TI DSP EMIF.

```
architecture AttributeUseSample_arch of AttributeUseSample  is
-------------------------------------------------------------------------
-- Component Instantiation
-------------------------------------------------------------------------
component ISERDES
generic (
  ......................
  ............
  .....
    INTERFACE_TYPE: string:= "MEMORY";
    IOBDELAY: string:= "NONE";
    IOBDELAY_TYPE: string:= "DEFAULT";
    IOBDELAY_VALUE: integer:= 0;
    NUM_CE: integer:= 2;
    SERDES_MODE: string:= "MASTER";
    SRVAL_Q1: bit:= '0';
    SRVAL_Q2: bit:= '0';
    SRVAL_Q3: bit:= '0';
    SRVAL_Q4: bit:= '0'
    );
port (
    O: out std_logic;
    Q1 : out std_logic;
  ........
  ............
  ...................
    SR : in std_logic
  );
-------------------------------------------------------------------------
-- Signal, Constants and Attributes Declarations
-------------------------------------------------------------------------
-- Constants
type TraceMatchVal is array (0 to NumberOfBits-1) of generic;
constant IdelayValue : TraceMatchValue := (
  4,-- First value
  6,
  2,
  0,
  ...
  ...-- Last value
  );

-- Signals
```

```
-- Attributes
-------------------------------------------------------------------------
---------------------------------
begin
.........................
.....................
.............
.......
...
DacIn : for n in 0 to NumberOfBits-1 generate   -- Data, Address or
Control.
  TheSerdes : ISERDES
    generic map (
        BITSLIP_ENABLE  => FALSE,
        DATA_RATE       => "SDR",
        DATA_WIDTH      => 4,
        INIT_Q1         => '0',
        INIT_Q2         => '0',
        INIT_Q3           => '0',
        INIT_Q4           => '0',
        INTERFACE_TYPE    => "MEMORY",
  -- The IDELAY is connected to the first FF.
        IOBDELAY          => "IFD",
        IOBDELAY_TYPE     => "FIXED",
  -- Value of the IDELAY coming from a list
        IOBDELAY_VALUE    => IdelayValue,
        NUM_CE            => 1,
        SERDES_MODE  => "MASTER",
        SRVAL_Q1     => '0',
        SRVAL_Q2     => '0',
        SRVAL_Q3     => '0',
        SRVAL_Q4     => '0'
        )
    port map (
        O  => ,
        Q1 => ,
        Q2 => ,
        Q3 => ,
      ..........
      ..............
      ......................
      ........................
end generate;

end  AttributeUseSample_arch ;
```

XILINX®

Appendix B

EMIF Register Field Descriptions

These tables are derived from the TMS320C64x datasheet.

Table B-1: **GBLCTL Register**

Bit	Field	SymVal	Description
31-20	Reserved		Reserved. The reserved bit location always reads as 0. A value written to this field has no effect.
19-18	EK2RATE		ECLKOUT2 rate. ECLKOUT2 runs at:
		FULLCLK	1× EMIF input clock (ECLKIN, CPU/4 clock, or CPU/6 clock) rate
		HALFCLK	1/2× EMIF input clock (ECLKIN, CPU/4 clock, or CPU/6 clock) rate
		QUARCLK	1/4× EMIF input clock (ECLKIN, CPU/4 clock, or CPU/6 clock) rate
		---	Reserved.
17	EK2HZ		ECLKOUT2 high-impedance control bit.
		CLK	ECLKOUT2 continues clocking during Hold (if EK2EN = 1).
		HIGHZ	ECLKOUT2 is in the high-impedance state during Hold.
16	EK2EN		ECLKOUT2 enable bit.
		DISABLE	ECLKOUT2 is held Low.
		ENABLE	ECLKOUT2 is enabled to clock.
15-14	Reserved		Reserved. The reserved bit location always reads as 0. A value written to this field has no effect.
13	BRMODE		Bus request mode (BRMODE) bit indicates if BUSREQ shows memory refresh status.
		MSTATUS	BUSREQ indicates memory access pending or in progress.
		MRSTATUS	BUSREQ indicates memory access or refresh pending or in progress.
12	Reserved		Reserved. The reserved bit location always reads as 0. A value written to this field has no effect.
11	BUSREQ		Bus request (BUSREQ) output bit indicates if the EMIF has an access/refresh pending or in progress.
		LOW	BUSREQ output is Low. No access/refresh pending.
		HIGH	BUSREQ output is High. Access/refresh pending or in progress.

Table B-1: **GBLCTL Register *(Continued)***

Bit	Field	SymVal	Description
10	ARDY		ARDY input bit. Valid ARDY bit is shown only when performing asynchronous memory access (when async $\overline{CE}$ is active).
		LOW	ARDY input is Low. External device is not ready.
		HIGH	ARDY input is High. External device is ready.
9	HOLD		HOLD input bit.
		LOW	HOLD input is Low. External device requesting EMIF.
		HIGH	HOLD input is High. No external request pending.
8	HOLDA		HOLDA output bit.
		LOW	HOLDA output is Low. External device owns EMIF.
		HIGH	HOLDA output is High. External device does not own EMIF.
7	NOHOLD		External NOHOLD enable bit.
		DISABLE	No Hold is disabled. Hold requests via the HOLD input are acknowledged via the HOLDA output at the earliest possible time.
		ENABLE	No Hold is enabled. Hold requests via the HOLD input are ignored.
6	EK1HZ		ECLKOUT1 high-impedance control bit.
		CLK	ECLKOUT1 continues clocking during Hold (if EK1EN = 1).
		HIGHZ	ECLKOUT1 is in high-impedance state during Hold.
5	EK1EN		ECLKOUT1 enable bit.
		DISABLE	ECLKOUT1 is held Low.
		ENABLE	ECLKOUT1 is enabled to clock.
4	CLK4EN		CLKOUT4 enable bit. The CLKOUT4 pin is muxed with the GP1 pin. Upon exiting reset, CLKOUT4 is enabled and clocking. After reset, CLKOUT4 can be configured as GP1 via the GPIO enable register (GPEN).
		DISABLE	CLKOUT4 is held High.
		ENABLE	CLKOUT4 is enabled to clock.
3	CLK6EN		CLKOUT 6 enable bit. The CLKOUT6 pin is muxed with the GP2 pin. Upon exiting reset, CLKOUT6 is enabled and clocking. After reset, CLKOUT6 can be configured as GP2 via the GPIO enable register (GPEN).
		DISABLE	CLKOUT6 is held High.
		ENABLE	CLKOUT6 is enabled to clock.
2-0	Reserved		Reserved. The reserved bit location always reads as 0. A value written to this field has no effect.

Table B-2: **CECTL Register**

Bit	Field	SymVal	Value	Description
31-28	WRSETUP	OF (value)	0x0-0xF	Write setup width. Number of clock cycles of setup time for address (EA), chip enable (CE), and byte enables (BE) before write strobe falls. For asynchronous read accesses, this is also the setup time of AOE before the read strobe (ARE) falls.
27-22	WRSTRB	OF (value)	0x0-0x3F	Write strobe width. The width of the write strobe (AWE) in clock cycles.
21-20	WRHLD	OF (value)	0x0-0x3	Write hold width. Number of clock cycles that address (EA) and byte strobes (BE) are held after write strobe rises. For asynchronous read accesses, this is also the hold time of AOE after ARE rises.
19-16	RDSETUP	OF (value)	0x0-0xF	Read setup width. Number of clock cycles of setup time for address (EA), chip enable (CE), and byte enables (BE) before read strobe falls. For asynchronous read accesses, this is also the setup time of AOE before ARE falls.
15-14	TA	OF (value)	0x0-0x3	Minimum Turnaround time. Turnaround time controls the minimum number of ECLKOUT cycles between a read followed by a write (same or different CE spaces), or between reads from different CE spaces. Applies only to asynchronous memory types.
13-8	RDSTRB	OF (value)	0x0-0x3	Read strobe width. The width of the read strobe (ARE) in clock cycles.
7-4	MTYPE		0x0-0xF	Memory type of the corresponding CE spaces.
		ASYNC8	0x0	8-bit asynchronous interface
		ASYNC16	0x1	16-bit asynchronous interface
		ASYNC32	0x2	32-bit asynchronous interface
		SDRAM32	0x3	32-bit SDRAM
		SYNC32	0x4	32-bit programmable synchronous memory
			0x5-0x7	Reserved
		SDRAM8	0x8	8-bit SDRAM
		SDRAM16	0x9	16-bit SDRAM
		SYNC8	0xA	8-bit programmable synchronous memory
		SYNC16	0xB	16-bit programmable synchronous memory
		ASYNC64	0xC	64-bit asynchronous interface
		SDRAM64	0xD	64-bit SDRAM
		SYNC64	0xE	64-bit programmable synchronous memory
			0xF	Reserved
3	WRHLDMSB	OF (value)		Write hold width MSB is the most-significant bit of write hold.
2-0	RDHLD	OF (value)		Read hold width. Number of clock cycles that address (EA) and byte strobes (BE) are held after read strobe rises. For asynchronous read accesses, this is also the hold time of AOE after ARE rising.

Table B-3: **CESEC Register**

Bit	Field	SymVal	Value	Description
31-7	Reserved		0	Reserved. The reserved bit location always reads as 0. A value written to this field has no effect.
6	SNCCLK			Synchronization clock selection bit.
		ECLKOUT1	0	Control/data signals for this CE space are synchronized to ECLKOUT1.
		ECLKOUT2	1	Control/data signals for this CE space are synchronized to ECLKOUT2.
5	RENEN			Read Enable enable bit.
		ADS	0	ADS mode. $\overline{SADS}$/$\overline{SRE}$ signal acts as the $\overline{SADS}$ signal. $\overline{SADS}$ goes active for reads, writes, and deselect. Deselect is issued after a command is completed if no new commands are pending from the EDMA (used for the SBSRAM or the ZBT SRAM interface).
		READ	1	Read enable mode. $\overline{SADS}$/$\overline{SRE}$ signal acts as the $\overline{SRE}$ signal. $\overline{SRE}$ goes Low only for reads. No deselect cycle is issued (used for the FIFO interface).
4	CEEXT			CE extension register ENABLE BIT.
		INACTIVE	0	CE goes inactive after the final command is issued (not necessarily when all the data has been latched).
		ACTIVE	1	On read cycles, the CE signal goes active when $\overline{SOE}$ goes active and stays active until $\overline{SOE}$ goes inactive. The $\overline{SOE}$ timing is controlled by SYNCRL (used for synchronous FIFO reads with glue, where CE gates $\overline{OE}$).
3-2	SYNCWL		0x0-0x3	Synchronous interface data write latency.
		0CYCLE	0x0	0 cycle read latency
		1CYCLE	0x1	1 cycle read latency
		2CYCLE	0x2	2 cycle read latency
		3CYCLE	0x3	3 cycle read latency
1-0	SYNCRL		0x0-0x3	Synchronous interface data read latency.
		0CYCLE	0x0	0 cycle read latency
		1CYCLE	0x1	1 cycle read latency
		2CYCLE	0x2	2 cycle read latency
		3CYCLE	0x3	3 cycle read latency

Table B-4: **PDTCTL Register**

Bit	Field	Value	Description
31-4	Reserved	0	Reserved. The reserved bit location always reads as 0. A value written to this field has no effect.
3-2	PDTWL	0x0-0x3	PDT write latency bits.
		0x0	$\overline{\text{PDT}}$ signal is asserted 0 cycles prior to the data phase of a write transaction.
		0x1	$\overline{\text{PDT}}$ signal is asserted 1 cycle prior to the data phase of a write transaction.
		0x2	$\overline{\text{PDT}}$ signal is asserted 2 cycles prior to the data phase of a write transaction.
		0x3	$\overline{\text{PDT}}$ signal is asserted 3 cycles prior to the data phase of a write transaction.
1-0	PDTRL	0x0-0x3	PDT read latency bits.
		0x0	$\overline{\text{PDT}}$ signal is asserted 0 cycles prior to the data phase of a read transaction.
		0x1	$\overline{\text{PDT}}$ signal is asserted 1 cycle prior to the data phase of a read transaction.
		0x2	$\overline{\text{PDT}}$ signal is asserted 2 cycles prior to the data phase of a read transaction.
		0x3	$\overline{\text{PDT}}$ signal is asserted 3 cycles prior to the data phase of a read transaction.

Table B-5: **SDCTL Register**

Bit	Field	SymVal	Value	Description
31	Reserved		0	Reserved. The reserved bit location always reads as 0. A value written to this field has no effect
30	SDBSZ			SDRAM bank size bit.
		2BANKS	0	One bank select pin (two banks)
		4BANKS	1	Two bank select pin (four banks)
29-28	SDRSZ		0x0-0x3	SDRAM row size bits.
		11ROW	0x0	11 row address pins (2048 rows per bank)
		12ROW	0x1	12 row address pins (4096 rows per bank)
		13ROW	0x2	13 row address pins (8192 rows per bank)
27-26	SDCSZ		0x0-0x3	SDRAM column size bits.
		9COL	0x0	9 column address pins (512 elements per row)
		8COL	0x1	8 column address pins
		10COL	0x2	10 column address pins
25	RFEN			Refresh enable bit.
		DISABLE	0	SDRAM refresh disabled
		ENABLE	1	SDRAM refresh enabled
24	INIT			Initialization bit.
		NO	0	No effect.
		YES	1	Initialize SDRAM in each CE space configured for SDRAM.
23-20	TRCD	(value)	0x0-0xF	Specifies the Trcd value of the SDRAM in EMIF clock cycles.
				TRCD = Trcd/Tcyc – 1

Table B-5: **SDCTL Register**

Bit	Field	SymVal	Value	Description
19-16	TRP	(value)	0x0-0xF	Specifies the Trp value of the SDRAM in EMIF clock cycles. TRP = Trp/Tcyc – 1
15-12	TRC	(value)	0x0-0xF	Specifies the Trc value of the SDRAM in EMIF clock cycles. TRC = Trc/Tcyc – 1
11-1	Reserved		0	Reserved. The reserved bit location always reads as 0. A value written to this field has no effect.
0	SLFRFR			Self-refresh mode, if SDRAM is used
		DISABLE	0	Self-refresh disabled
		ENABLE	1	Self-refresh enabled
				If SDRAM is not used:
		DISABLE	0	General-purpose output, SDCKE = 1
		ENABLE	1	General-purpose output, SDCKE = 0

Table B-6: **SDTIM Register**

Bit	Field	SymVal	Value	Description
31-26	Reserved		0	Reserved. The reserved bit location always reads as 0. A value written to this field has no effect.
25-24	XRFR	(value)	0x0-0x3	Extra refreshes controls the number of refreshes performed to the SDRAM when the refresh counter expires.
23-12	CNTR	(value)	0x0-0xFFF	Current value of the refresh counter.
11-0	PERIOD	(value)	0x0-0xFFF	Refresh period in EMIF clock cycles.

Table B-7: **SDEXT Register**

Bit	Field	SymVal	Value	Description
31-21	Reserved		0	Reserved. The reserved bit location always reads as 0. A value written to this field has no effect
20	WR2RD	(value)	0-1	Minimum number of cycles between WRITE and READ commands of the SDRAM in ECLKOUT cycles. WR2RD = (# of cycles WRITE to READ) - 1
19-18	WR2DEAC	(value)	0x0-0x3	Minimum number of cycles between WRITE and DEAC/DCAB commands of the SDRAM in ECLKOUT cycles. WR2DEAC = (# of cycles WRITE to DEAC/DCAB) - 1
17	WR2WR	(value)	0x0-0x1	Minimum number of cycles between WRITE commands of the SDRAM in ECLKOUT cycles. WR2WR = (# of cycles WRITE to WRITE) – 1

Table B-7: **SDEXT Register *(Continued)***

Bit	Field	SymVal	Value	Description
16-15	R2WDQM	(value)	0x0-0x3	Number of cycles that BEx must be High preceding a WRITE interrupting a READ. R2WRDQM = (# of cycles BEx High) – 1
14-12	RD2WR	(value)	0x0-0x3	Number of cycles between READ to WRITE commands of the SDRAM in ECLKOUT cycles. RD2WR = (# of cycles READ to WRITE) – 1
11-10	RD2DEAC	(value)	0x0-0x3	Number of cycles between READ and DEAC/DCAB commands of the SDRAM in ECLKOUT cycles. RD2DEAC = (# of cycles READ to DEAC/DCAB) – 1
9	RD2RD	(value)	 0 1	Number of cycles between READ commands. READ to READ = 1 ECLKOUT cycle READ to READ = 2 ECLKOUT cycles
8-7	THZP	(value)	0x0-0x3	Thzp (also Troh) value of the SDRAM in ECLKOUT cycles. THZP = Thzp / Tcyc – 1
6-5	TWR	(value)	0x0-0x3	Twr value of the SDRAM in ECLKOUT cycles. TWR = Twr / Tcyc – 1
4	TRRD	(value)	 0 1	Trrd value of the SDRAM in ECLKOUT cycles. TRRD = 2 ECLKOUT cycles TRRD = 3 ECLKOUT cycles
3-1	TRAS	(value)	0x0-0x7	Tras values of the SDRAM in ECLKOUT cycles. TRAS = Tras / Tcyc – 1
0	TCL	(value)	 0 1	CAS latency of the SDRAM in ECLKOUT cycles. CAS latency = 2 ECLKOUT cycles CAS latency = 3 ECLKOUT cycles

XILINX®

Appendix C

Related References

This appendix provides supplementary material useful with this application note.

Datasheets and User Guides

Xilinx Documents

- Virtex-II Platform FPGAs: Complete Data Sheet (DS031)
- Virtex-II Platform FPGA User Guide (UG002)
- Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet (DS083)
- Virtex-II Pro Platform FPGA: User Guide (UG012)
- Spartan-3 FPGA Family: Complete Data Sheet (DS099)
- Virtex-4 Family Overview (DS112)
- Virtex-4 User Guide (UG070)

Texas Instruments Documents

- TMS320C6000 DSP External Memory Interface Reference Guide (SPRU266b)
- TMS320C6416 Fixed-Point Digital Signal Processors datasheet (SPRS226e)
- TMS320C64x Technical Overview (SPRU395b)
- TMS320C6000 EMIF to External Asynchronous SRAM Interface (SPRA542)
- TMS320C6000 EMIF to External Flash Memory (SPRA568a)
- TMS320C6000 DSP Peripherals Overview Reference Guide (SPRU190)
- TMS320C6201 Fixed-Point Digital Signal Processor (SPRS051)
- Using IBIS Models for Timing Analysis (SPRA839a)

PCB and Design Guides

- Xilinx XAPP623: "Power Distribution System (PDS) Design: Using Bypass/Decoupling Capacitors"
- *RF Circuit Design* (theory and applications): Reinhold Ludwig/Pavel Bretchko (Prentice Hall ISBN 0-13-095323-7)
- RFI/EMI/EMC: *A Designer's Handbook*: Gary A. Breed
- *Microwave Circuit Analysis and Amplifier Design*: Samuel Y. Liao (Prentice Hall ISBN 0-13-586736-3)

- *Kluwer Technische Boeken*: Electronica Vademecum: Kluwer Deventer-Antwerpen, 1980
- *Reference Data for Engineers*: Edward C. Jordan (Howard W. Sams & Co.)
- *Transmission Lines for Digital and Communications Networks*: Richard E. Matick (McGraw Hill, 1969)
- IPC publications: IPC-2141 and IPC-D-317A, along with corrections and adjustments to these publications from IPC and others
- *Radio Handbook*, 23rd edition: William I. Orr (Newnes, 1997. ISBN 0-7506-9947-7)
- *High-Speed Digital Design*: Howard Johnson – Martin Graham (Prentice Hall ISBN 0-13-395724-1)
- *High-Speed Signal Propagation*: Howard Johnson – Martin Graham (Prentice Hall ISBN 0-13-084408-x)