XILINX®

XAPP463 (v2.0) March 1, 2005

Using Block RAM in Spartan-3 Generation FPGAs

Summary

For applications requiring large, on-chip memories, Spartan™-3 Generation FPGAs provides plentiful, efficient SelectRAM™ memory blocks. Using various configuration options, SelectRAM blocks create RAM, ROM, FIFOs, large look-up tables, data width converters, circular buffers, and shift registers, each supporting various data widths and depths. This application note describes the features and capabilities of block SelectRAM and illustrates how to specify the various options using the Xilinx CORE Generator™ system or via VHDL or Verilog instantiation. Various non-obvious block RAM applications are discussed with references to additional tools, application notes, and documentation.

Introduction

All Spartan-3 Generation FPGAs, including Spartan-3, Spartan-3L, and Spartan-3E devices, feature multiple block RAM memories, organized in columns. The total amount of block RAM memory depends on the size of the Spartan-3 Generation FPGA. Table 1 shows the available block RAM for Spartan-3 and Spartan-3L FPGAs.

Table 1: **Block RAM Available in Spartan-3 and Spartan-3L Devices**

Device	RAM Columns	RAM Blocks Per Column	Total RAM Blocks	Total RAM Bits	Total RAM Kbits
XC3S50	1	4	4	73,728	72K
XC3S200	2	6	12	221,184	216K
XC3S400	2	8	16	294,912	288K
XC3S1000/L	2	12	24	442,368	432K
XC3S1500/L	2	16	32	589,824	576K
XC3S2000	2	20	40	737,280	720K
XC3S4000/L	4	24	96	1,769,472	1,728K
XC3S5000	4	26	104	1,916,928	1,872K

Notes:

1. 1Kbit = 1,024 bits, per memory conventions.

Table 3: **SelectRAM 18K Block Memory Features and Applications**

Total RAM bits, including parity	18,432 (16K data + 2K parity)
Memory Organizations	16Kx1 8Kx2 4Kx4 2Kx8 (no parity) 2Kx9 (x8 + parity) 1Kx16 (no parity) 1Kx18 (x16 + 2 parity) 512x32 (no parity) 512x36 (x32 + 4 parity) 256x72 (single-port only)
Parity	Available and optional only for organizations greater than byte-wide. Parity bits optionally available as extra data bits.
Performance	200 MHz (estimated)
Timing Interface	Simple synchronous interface. Similar to reading and writing from a register with a setup time for write operations and clock-to-output delay for read operations.
Single-Port	Yes
True Dual-Port	Yes
ROM, Initial RAM Contents	Yes
Mixed Data Port Widths	Yes
Power-Up Condition	User-defined data, defaults to zero
Potential Applications	Local data storage, FIFOs, elastic stores, register files, buffers, stacks, circular buffers, shift registers, delay lines, waveform storage and generation, direct digital synthesis, CAMs, associative memories, function tables, function generators, wide logic functions, code converters, encoders, decoders, counters, state machines, microsequencers, program storage for embedded processor(s)

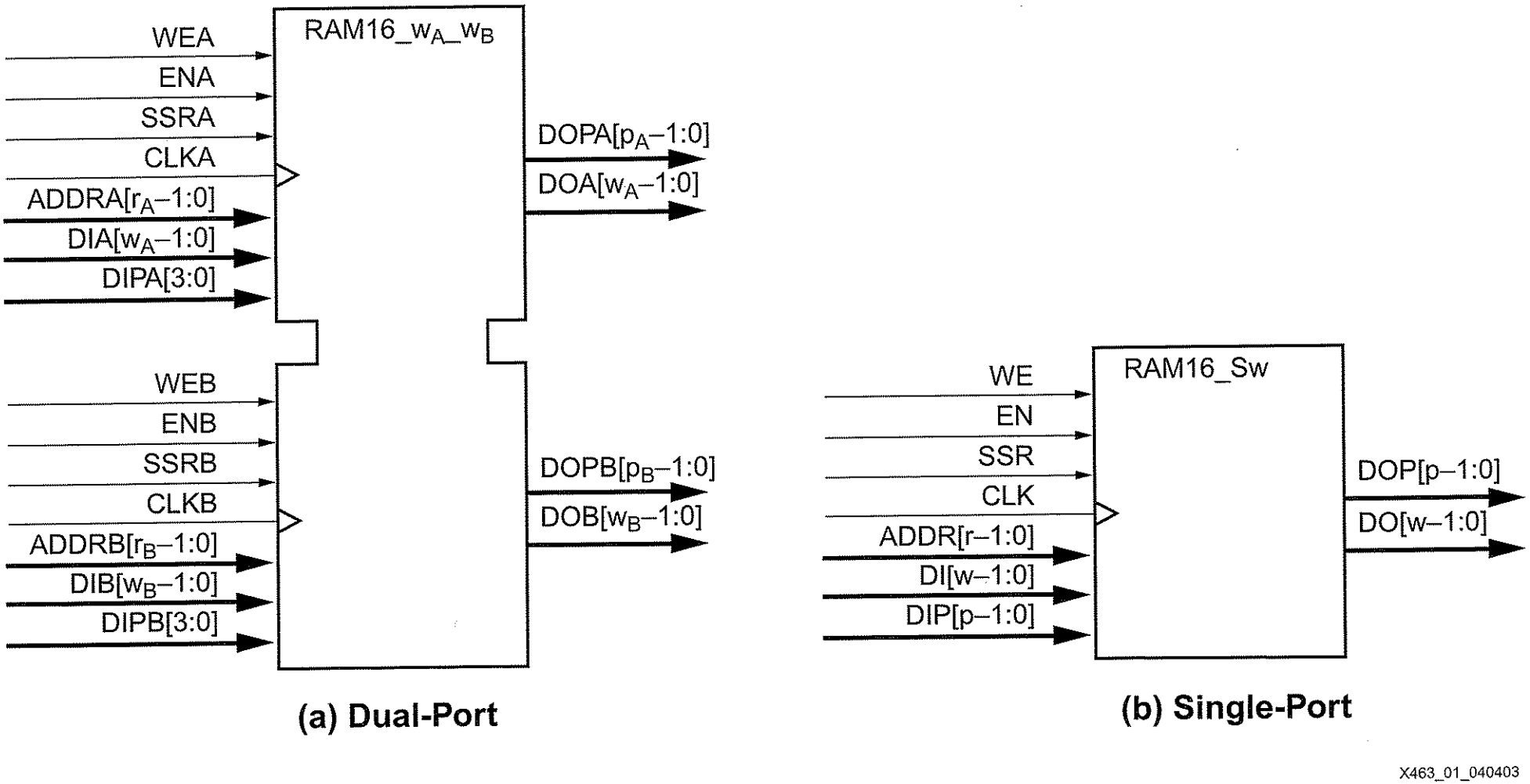
locations are updated with the input data during a simultaneous Write operation. When WRITE_MODE is READ_FIRST, the output latches are updated with the data previously stored in the addressed RAM location and the new data on the DI and DIP inputs is stored at the address RAM location. When WRITE_MODE is NO_CHANGE, the data output latches are unaffected by a simultaneous Write operation and retain their present state.

General Characteristics

- A write operation requires only one clock edge.
- A read operation requires only one clock edge.
- All inputs are registered with the port clock and have a setup-to-clock timing specification.
- All outputs have a read-through function or one of three read-during-write functions, depending on the state of the WE pin. The outputs relative to the port clock are available after the clock-to-out timing interval.
- Block RAM cells are true synchronous RAM memories and do not have a combinatorial path from the address to the output.
- The ports are completely independent of each other without arbitration. Each port has its own clocking, control, address, read/write functions, initialization, and data width.
- Output ports are latched with a self-timed circuit, guaranteeing glitch-free read operations. The state of the output port does not change until the port executes another read or write operation.

Functional Compatibility with Other Xilinx FPGA Families

The block RAM on Spartan-3 Generation FPGAs is functionally identical to block RAM on the Xilinx Virtex-II/Virtex-II Pro FPGA families. Consequently, design tools that support Virtex-II and Virtex-II Pro block RAM also support with Spartan-3 Generation FPGAs.



Notes:

1. w_A and w_B are integers representing the total data path width (i.e., data bits plus parity bits) at ports A and B, respectively.
2. p_A and p_B are integers that indicate the number of data path lines serving as parity bits.
3. r_A and r_B are integers representing the address bus width at ports A and B, respectively.
4. The control signals CLK, WE, EN, and SSR on both ports have the option of inverted polarity.

Figure 1: **SelectRAM 18K Blocks Perform as Dual-Port (a) and Single-Port (b) Memory**

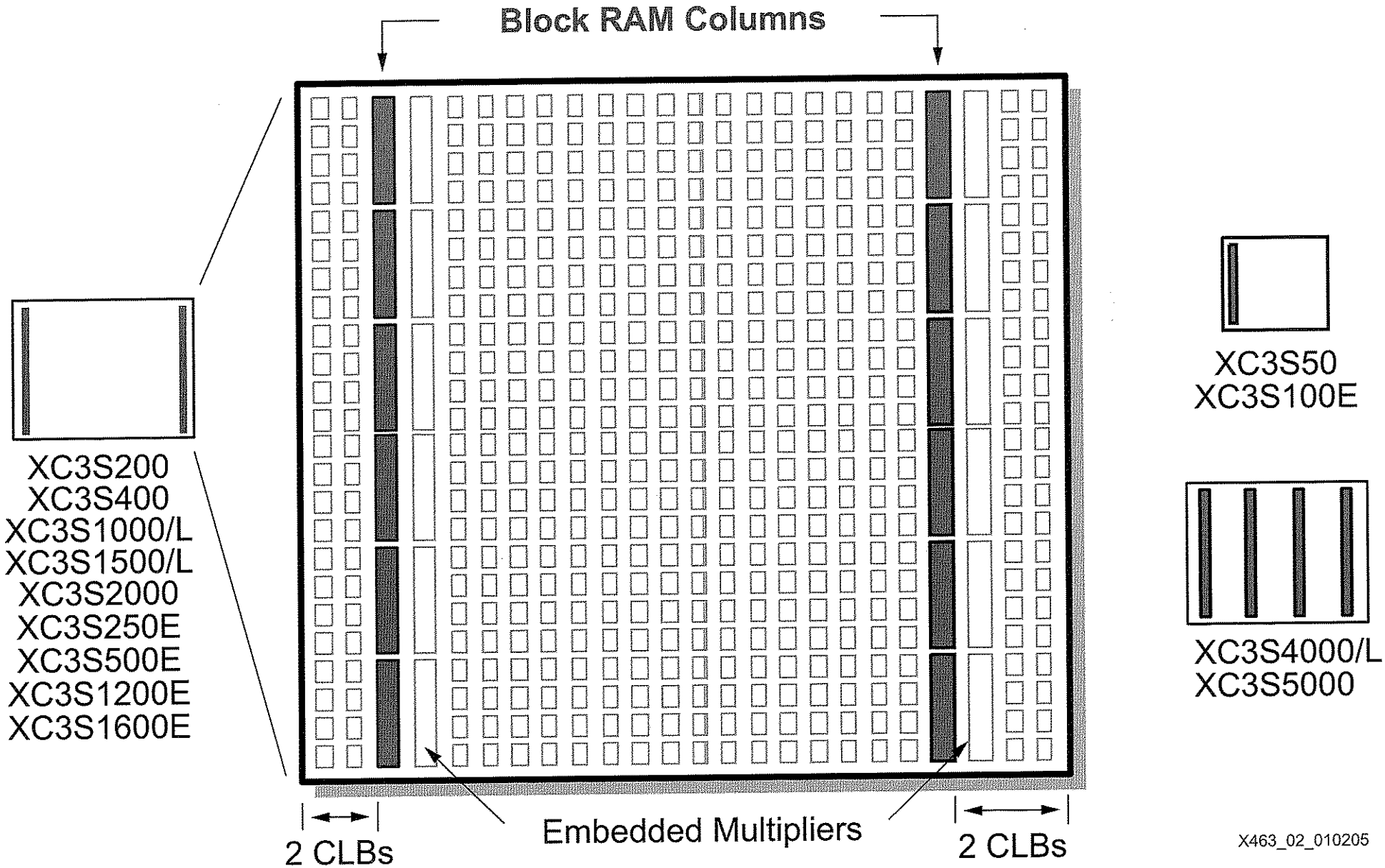


Figure 2: **Block RAMs Arranged in Columns with Detailed Floorplan of XC3S200**

Immediately adjacent to each block RAM is an embedded 18x18 hardware multiplier. Co-locating block RAM and the embedded multipliers improves the performance of some digital signal processing functions.

Immediately adjacent to each block RAM is an embedded 18x18 hardware multiplier. Co-locating block RAM and the embedded multipliers improves the performance of some digital signal processing functions.

Special interconnect surrounding the block RAM provides efficient signal distribution for address and data. Furthermore, special provisions allow multiple block RAMs to be cascaded to create wider or deeper memories.

Spartan-3 block RAM is constructed of true dual-port memory and simultaneously supports all the data flows and operations shown in Figure 3. Both ports access the same set of memory bits but with two potentially different address schemes depending on the port's data width.

1. Port A behaves as an independent single-port RAM supporting simultaneous read and write operations using a single set of address lines.
2. Port B behaves as an independent single-port RAM supporting simultaneous read and write operations using a single set of address lines.
3. Port A is the write port with a separate write address and Port B is the read port with a separate read address. The data widths for Port A and Port B can be different also.
4. Port B is the write port with a separate write address and Port A is the read port with a separate read address. The data widths for Port B and Port A can be different also.

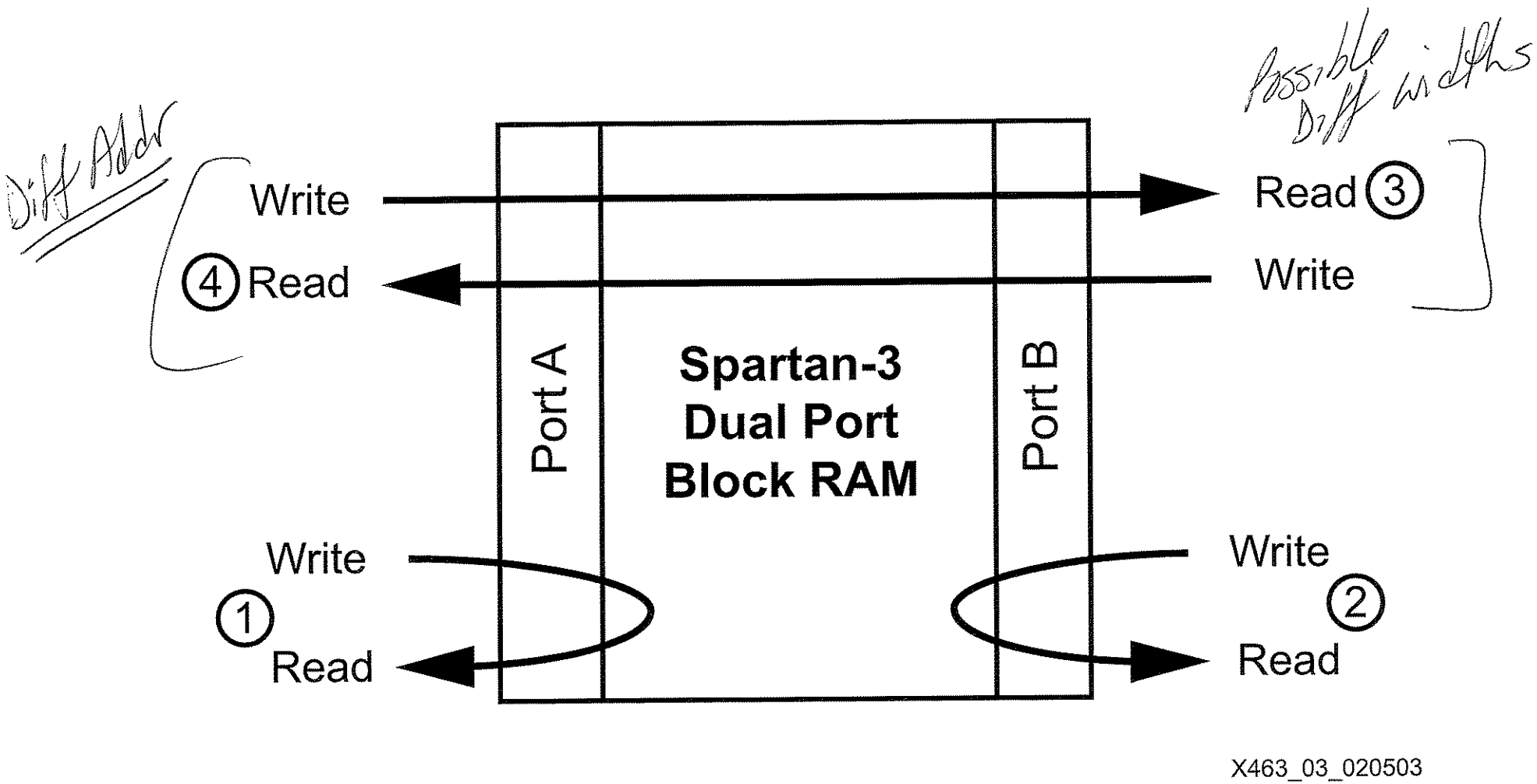


Figure 3: **Block RAM Support Single- and Dual-Port Data Transfers**

ls connected to a block RAM primitive divide into four categories, as listed l
ts the block RAM interface signals, the signals names for both single-port a

1. Data Inputs and Outputs
2. Parity Inputs and Outputs, available when a data port is byte-wide or wider
3. Address inputs to select a specific memory location
4. Various control signals that manage read, write, or set/reset operations.

Table 4: **Block RAM Interface Signals**

Signal Description	Single Port	Dual Port		Direction
		Port A	Port B	
Data Input Bus	DI	DIA	DIB	Input
Parity Data Input Bus (available only for byte-wide and wider organizations)	DIP	DIPA	DIPB	Input
Data Output Bus	DO	DOA	DOB	Output
Parity Data Output (available only for byte-wide and wider organizations)	DOP	DOPA	DOPB	Output
Address Bus	ADDR	ADDRA	ADDRB	Input
Write Enable	WE	WEA	WEB	Input
Clock Enable	EN	ENA	ENB	Input
Synchronous Set/Reset output latches	SSR	SSRA	SSRB	Input
Clock	CLK	CLKA	CLKB	Input

Read Behavior During Simultaneous Write — WRITE_MODE

To maximize data throughput and utilization of the dual-port memory at each clock edge, block RAM memory supports one of three write modes for each memory port. These different modes determine which data is available on the output latches after a valid write clock edge to the same port. The default mode, WRITE_FIRST, provides backwards compatibility with the older Virtex™/Virtex-E and Spartan-IIE FPGA architectures and is also the default behavior for Virtex-II/Virtex-II Pro™ devices. However, READ_FIRST mode is the most useful as it increases the efficiency of block RAM memory at each clock cycle, allowing designs to use maximum bandwidth. In READ_FIRST mode, a memory port supports simultaneous read and write operations to the same address on the same clock edge, free of any timing complications.

Table 9 outlines how the WRITE_MODE setting affects the output data latches on the same port, and how it affects the output latches on the opposite port during a simultaneous access to the same address.

Table 9: **WRITE_MODE Affects Data Output Latches During Write Operations**

Write Mode	Effect on Same Port	Effect on Opposite Port (dual-port mode only, same address)
WRITE_FIRST Read After Write (Default)	Data on DI, DIP inputs written into specified RAM location and simultaneously appears on DO, DOP outputs.	Invalidates data on DO, DOP outputs. ("Data" State")
READ_FIRST Read Before Write (Recommended)	Data from specified RAM location appears on DO, DOP outputs. Data on DI, DIP inputs written into specified location.	Data from specified RAM location appears on DO, DOP outputs.
NO_CHANGE No Read on Write	Data on DO, DOP outputs remains unchanged. Data on DI, DIP inputs written into specified location.	Invalidates data on DO, DOP outputs. (Mismatch between ports)

Mode selection is set by configuration. One of these three modes is set individually for each port by an attribute. The default mode is WRITE_FIRST.

Mode selection is set by configuration. One of these three modes is set individually for each port by an attribute. The default mode is WRITE_FIRST.

WRITE_FIRST or Transparent Mode (Default)

The WRITE_FIRST mode is the default operating mode for backward compatibility reasons. For new designs, READ_FIRST mode is recommended.

In this mode, the input data is written into the addressed RAM location memory and simultaneously stored in the data output latches, resulting in a transparent write operation, as shown in Figure 11. The WRITE_FIRST mode provides backwards compatibility with the 4K-bit blocks RAMs on Virtex/Virtex-E and Spartan-II/Spartan-IIE FPGAs and is also the default mode for Virtex-II/Virtex-II Pro block RAMs.

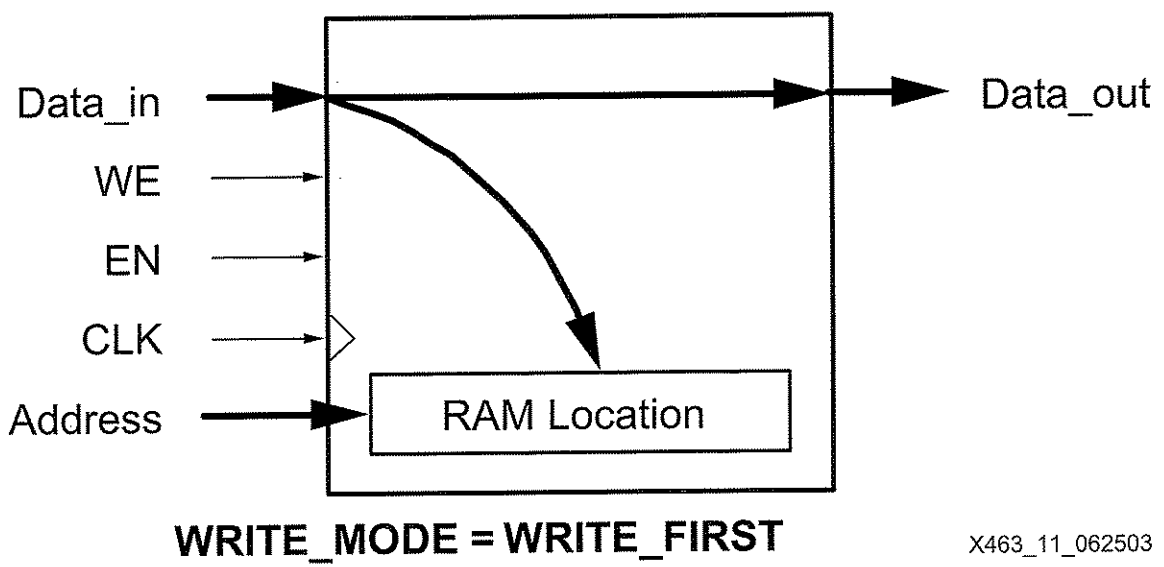


Figure 11: **Data Flow during a WRITE_FIRST Write Operation**

Figure 12 demonstrates that a valid write operation during a valid read operation results in the write data appearing on the data output.

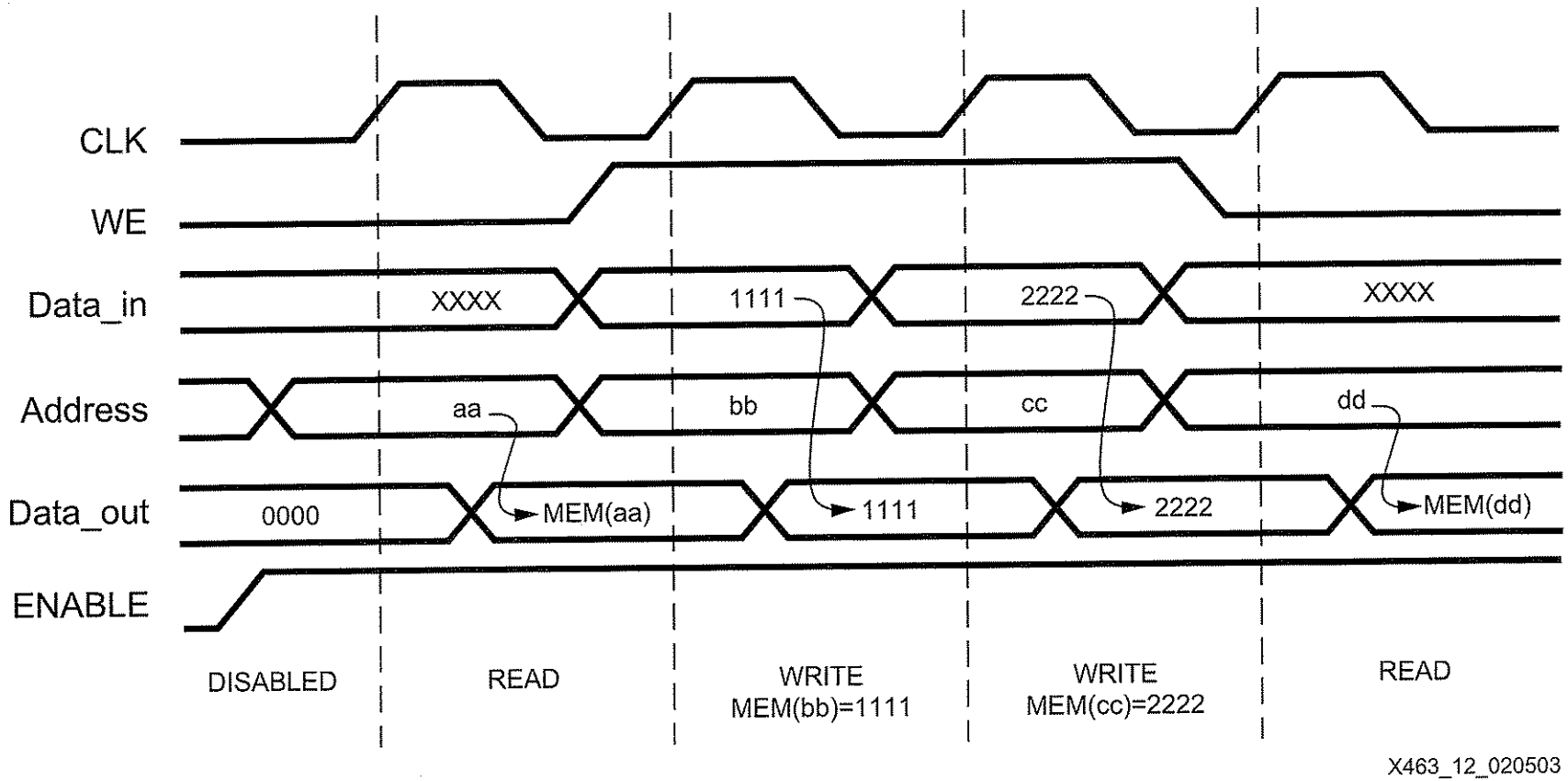


Figure 12: **WRITE_FIRST Mode Waveforms**

READ_FIRST or Read-Before-Write Mode

In READ_FIRST mode, data previously stored at the write address appears on the output latches, while the new input data is stored in memory, resulting in a read-before-write operation

READ_FIRST or Read-Before-Write Mode

In READ_FIRST mode, data previously stored at the write address appears on the output latches, while the new input data is stored in memory, resulting in a read-before-write operation shown in Figure 13. The older RAM data appears on the data output while the new RAM data is stored in the specified RAM location. READ_FIRST mode is the recommended operating mode.

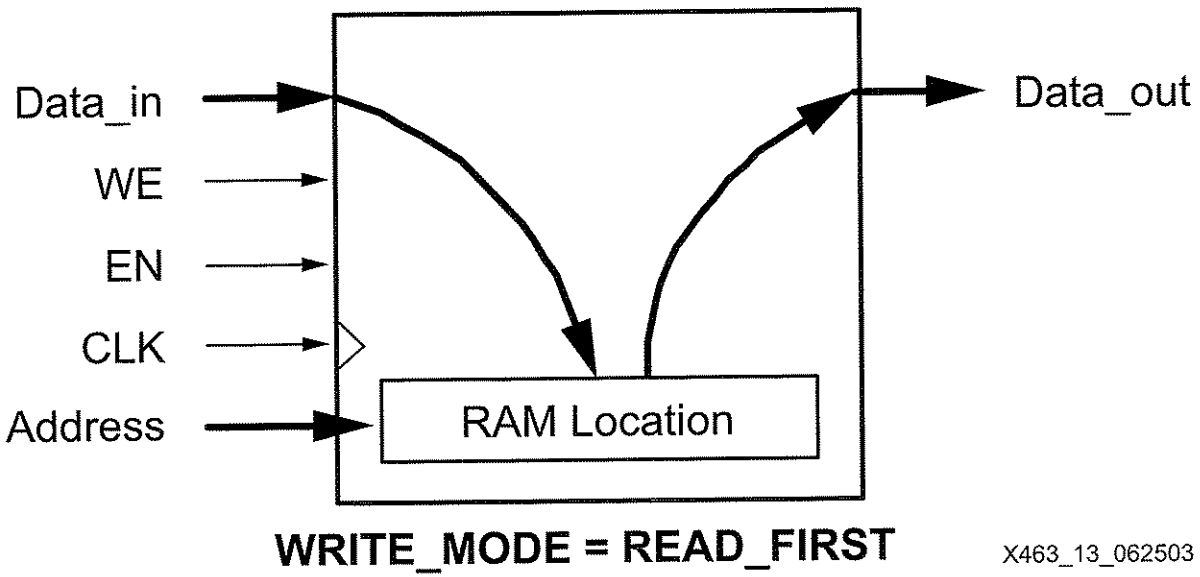


Figure 13: **Data Flow during a READ_FIRST Write Operation**

Figure 14 demonstrates that the older RAM data always appears on the data output, regardless of a simultaneous write operation.

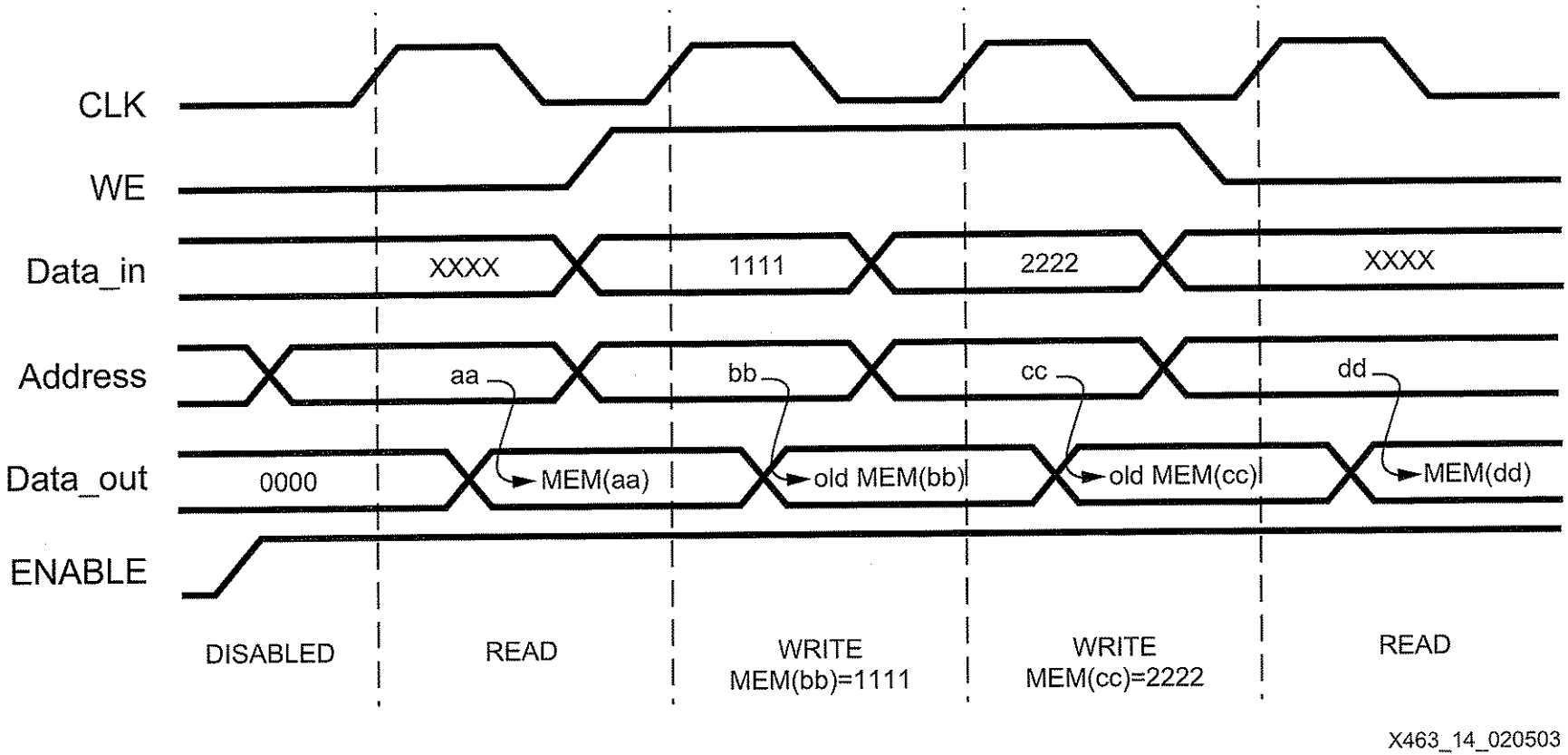


Figure 14: **READ_FIRST Mode Waveforms**

This mode is particularly useful for building circular buffers and large, block-RAM-based shift registers. Similarly, this mode is useful when storing FIR filter taps in digital signal processing applications. Old data is copied out from RAM while new data is written into RAM.

NO_CHANGE Mode

In NO_CHANGE mode, the output latches are disabled and remain unchanged during a

NO_CHANGE Mode

In NO_CHANGE mode, the output latches are disabled and remain unchanged during a simultaneous write operation, as shown in Figure 15. This behavior mimics that of simple synchronous memory where a memory location is either read or written during a clock cycle, but not both.

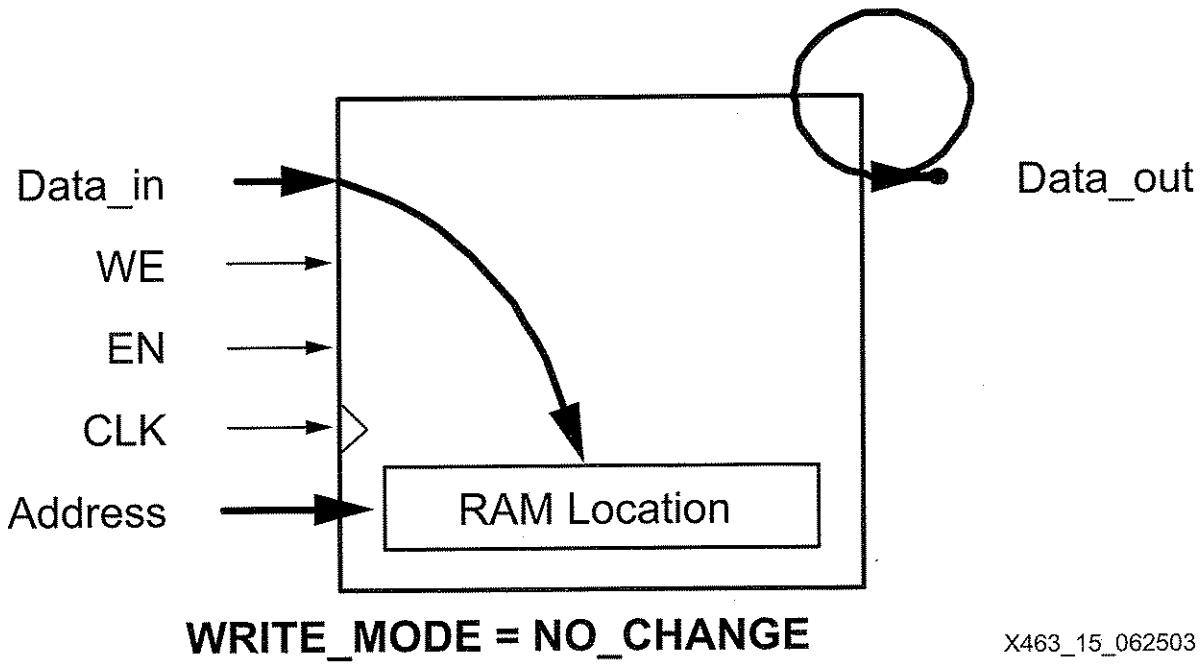


Figure 15: **Data Flow during a NO_CHANGE Write Operation**

The NO_CHANGE mode is useful in a variety of applications, including those where the block RAM contains waveforms, function tables, coefficients, and so forth. The memory can be updated without affecting the memory output.

Figure 16 shows that the data output retains the last read data if there is a simultaneous write operation on the same port.

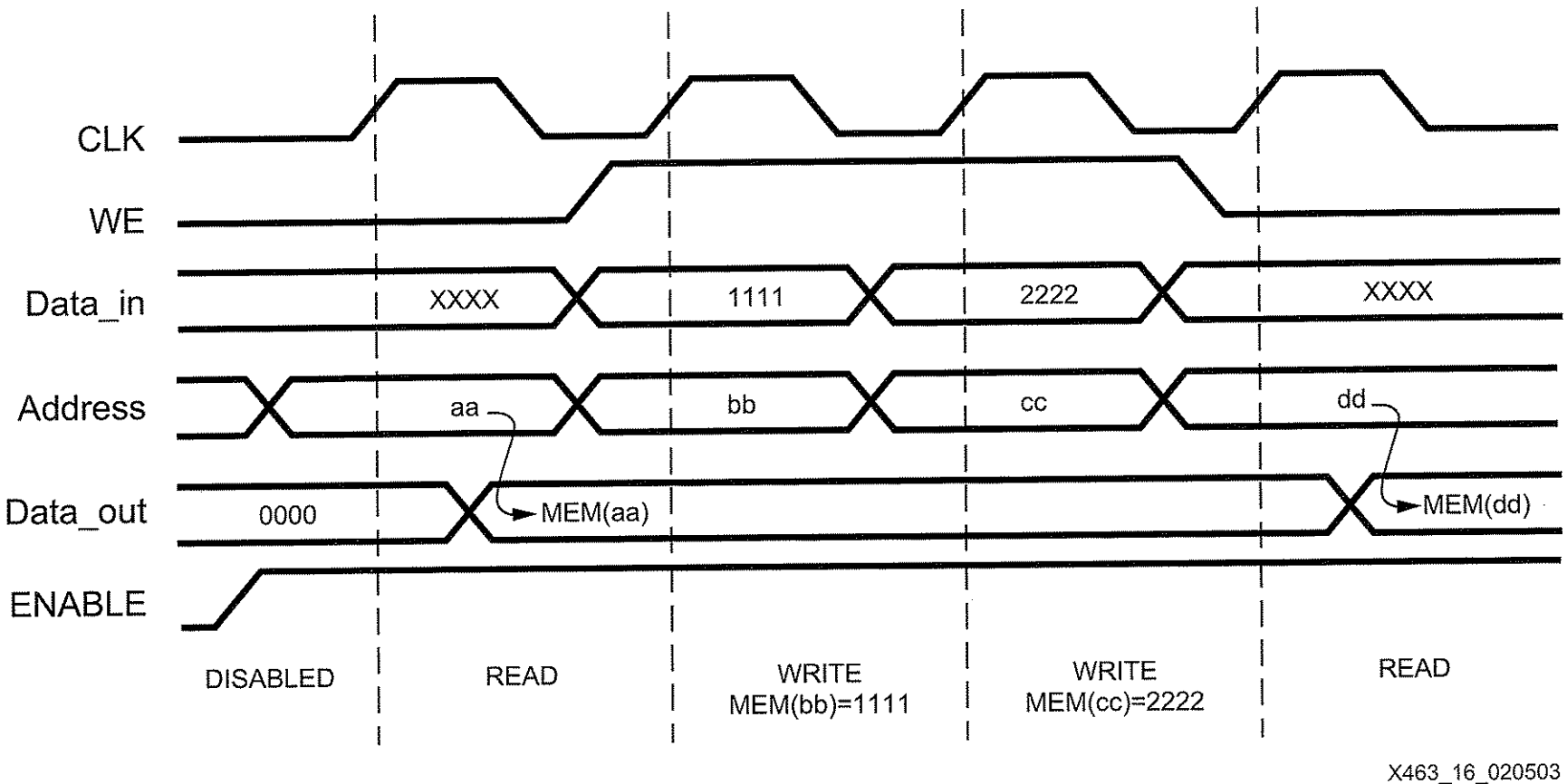


Figure 16: **NO_CHANGE Mode Waveforms**

CORE Generator System — Write Mode

To specify the WRITE_MODE in the CORE Generator system, locate the settings for Write Mode as shown in Figure 17. Select between Read After Write (WRITE_FIRST), Read Before Write (READ_FIRST) or No Read On Write (NO_CHANGE).

Simultaneous Write and Synchronous Set/Reset Operations

If a simultaneous write operation occurs during the synchronous set/reset operation, then the data on the DI and DIP inputs is stored at the RAM location specified by the ADDR input. However, the data output latches are initialized to the SRVAL attribute value as described immediately above.

Read Operations Occur on Every Clock Edge When Enable is Asserted

Read operations are synchronous and require a clock edge and an asserted clock enable. The data output behavior depends on whether or not a simultaneous write operation occurs during the read cycle.

If no simultaneous write cycle occurs during a valid read cycle, then the read address is registered on the read port and the data stored in RAM at that address is simply loaded into the output latches after the RAM access interval passes.

However, if there is a simultaneous write cycle during the read cycle, then the output behavior depends on which of the three write modes is selected, as described immediately below.

Write Operations Always Have Simultaneous Read Operation, Data Output Latches Affected

During a Write operation, a simultaneous Read operation occurs. The WRITE_MODE attribute determines the behavior of the data output latches during the Write operation (refer to Read Behavior During Simultaneous Write — WRITE_MODE, page 14). By default, WRITE_MODE is WRITE_FIRST and the data output latches and the addressed RAM locations are updated with the input data during a simultaneous Write operation. When WRITE_MODE is READ_FIRST, the output latches are updated with the data previously stored in the addressed RAM location and the new data on the DI and DIP inputs is stored at the address RAM location. When WRITE_MODE is NO_CHANGE, the data output latches are unaffected by a simultaneous Write operation and retain their present state.

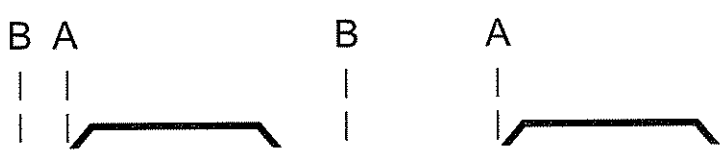
As a dual-port RAM, the block RAM allows both ports to simultaneously access the same memory cell. Potentially, conflicts arise under the following conditions.

1. If the clock inputs to the two ports are asynchronous, then conflicts occur if clock-to-clock setup time requirements are violated.
2. Both memory ports write different data to the same RAM location during a valid write cycle.
3. If a port uses WRITE_MODE=NO_CHANGE or WRITE_FIRST, a write to the port invalidates the read data output latches on the opposite port.

If Port A and Port B different memory organizations and consequently different widths, only the overlapping bits are invalid when conflicts occur.

Timing Violation Conflicts

When one port writes to a given memory cell, the other port must not address that memory cell—either for a write or a read operation—within the clock-to-clock setup window, which is equivalent to the block RAM minimum clock period (Tbpwh + Tbpwl), specified in the Spartan-3 data sheet. Figure 19 describes this situation where both ports operate from asynchronous clock inputs.



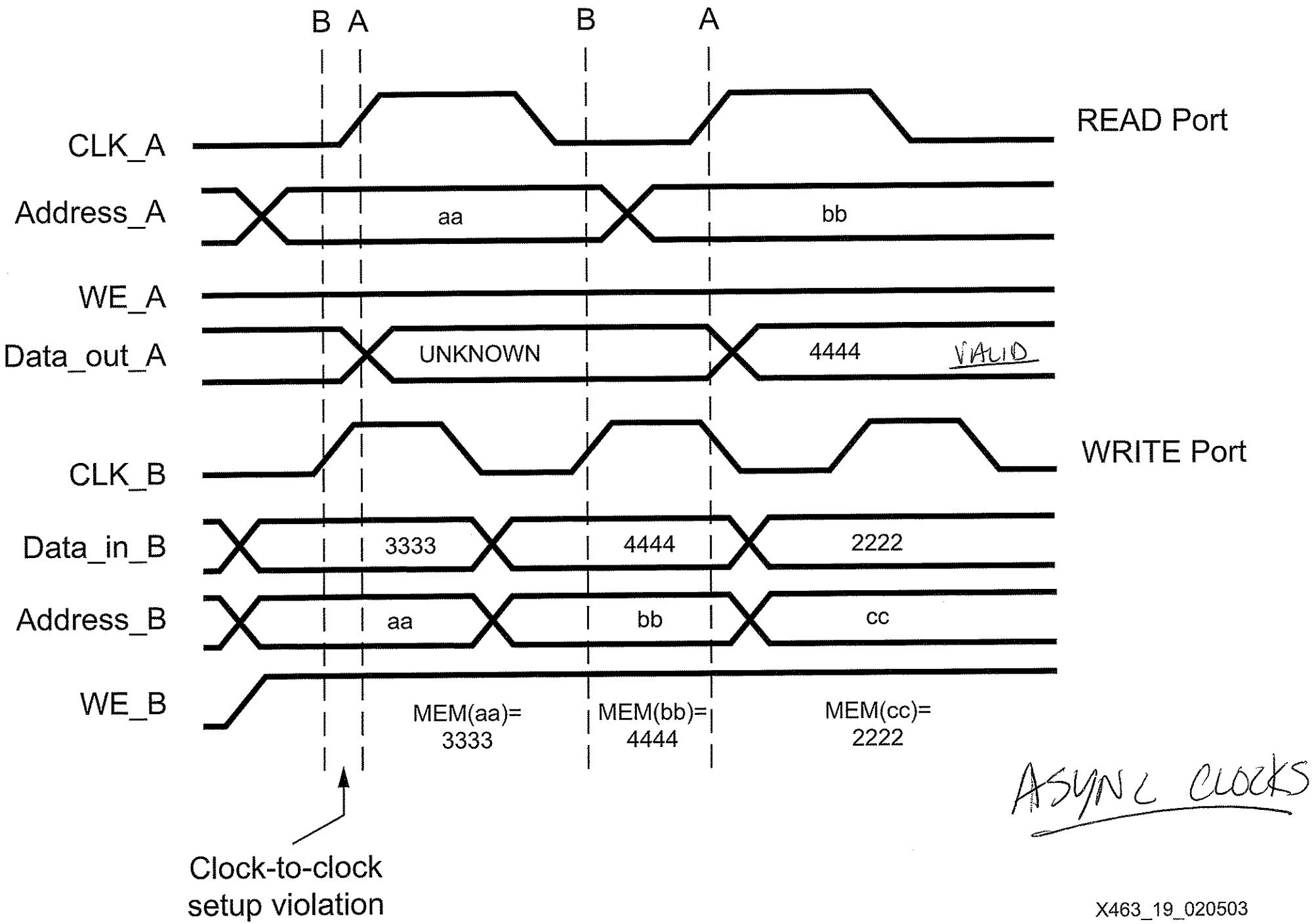


Figure 19: **Clock-to-Clock Timing Conflicts**

; byte-writes, memory contents are not corrupted when separate bytes are writ
vord. RAM contents are corrupted only when both ports attempt to write the s
ustrates this case. Assume ADDRA = ADDRB = 0.

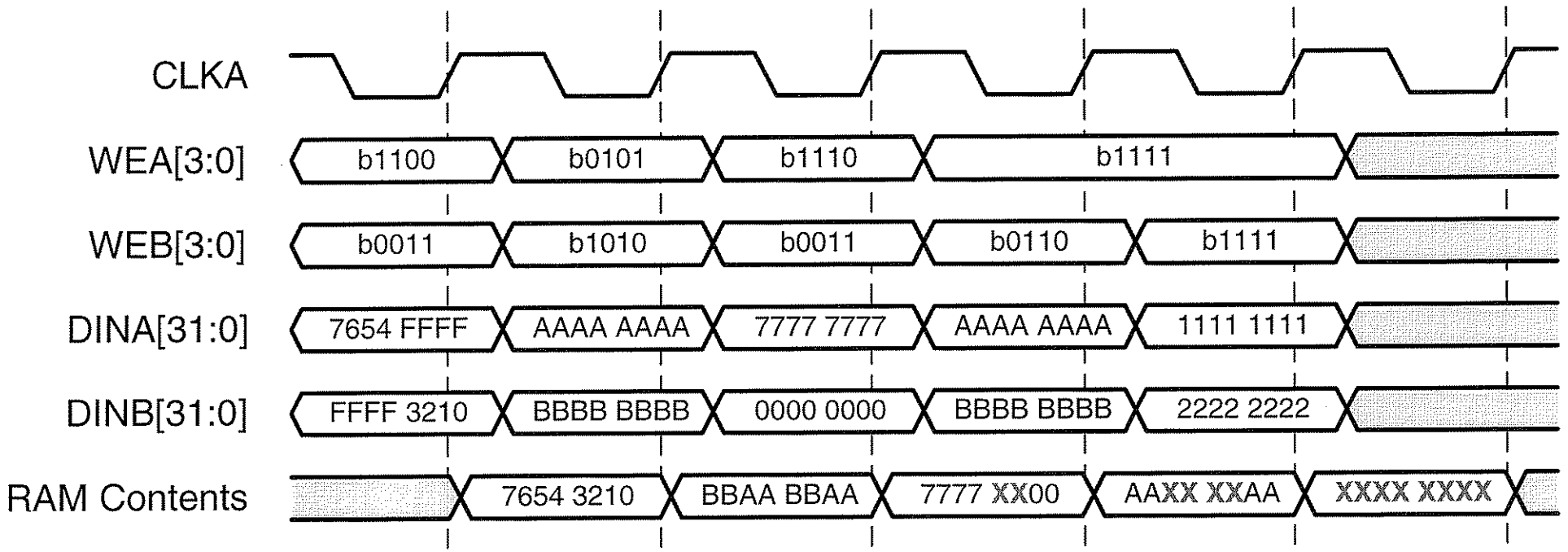


Figure 14: **Write-Write Collision Example**

SYNC Clocks

s Write-Read Collisions

BYTE ENB

ous write-read collision may occur if a port attempts to write a memory locatic
eads the same location. While memory contents are not corrupted in write-read
of the output data depends on the write port operating mode

memory. Write operations always succeed and the write port's output data latches behave as described by the port's WRITE_MODE attribute. If the write port is configured with WRITE_MODE set to NO_CHANGE or WRITE_FIRST, then a write operation to the port invalidates the data output latches on the opposite port, as shown in Table 12.

Using the READ_FIRST mode does not cause conflicts on the opposite port.

Table 12: **Conflicts to Output Latches Based on WRITE_MODE**

Input Signals								Output Signals			
Port A				Port B				Port A		Port B	
WEA	**CLKB**	**DIPA**	**DIA**	**WEB**	**CLKA**	**DIPB**	**DIB**	**DOPA**	**DOA**	**DOPB**	**DOB**
WRITE_MODE_A=NO_CHANGE											
1	↑	DIPA	DIA	0	↑	DIPB	DIB	No Chg	No Chg	?	?
WRITE_MODE_B=NO_CHANGE											
0	↑	DIPA	DIA	1	↑	DIPB	DIB	?	?	No Chg	No Chg
WRITE_MODE_A=WRITE_FIRST											
1	↑	DIPA	DIA	0	↑	DIPB	DIB	DIPA	DOA	?	?
WRITE_MODE_B=WRITE_FIRST											
0	↑	DIPA	DIA	1	↑	DIPB	DIB	?	?	DIPB	DIB
WRITE_MODE_A=WRITE_FIRST, WRITE_MODE_B=WRITE_FIRST											
0	↑	DIPA	DIA	1	↑	DIPB	DIB	?	?	?	?

Notes:

1. ADDRA=ADDRB, ENA=1, ENB=1, ?=Unknown or invalid data

Conflict Resolution

There is no dedicated monitor to arbitrate the result of identical addresses on both ports. The application must time the two clocks appropriately. However, conflicting simultaneous writes to the same location never cause any physical damage.