Read Operation

In latch mode, the read operation uses one clock edge. The read address is registered on the read port, and the stored data is loaded into the output latches after the RAM access time. When using the output register, the read operation will take one extra latency cycle to arrive at the output.

Write Operation

A write operation is a single clock-edge operation. The write address is registered on the write port, and the data input is stored in memory.

Write Modes

Three settings of the write mode determines the behavior of the data available on the output latches after a write clock edge: WRITE_FIRST, READ_FIRST, and NO_CHANGE. The Write mode attribute can be individually selected for each port. The default mode is WRITE_FIRST. WRITE_FIRST outputs the newly written data onto the output bus. READ_FIRST outputs the previously stored data while new data is being written. NO_CHANGE maintains the output previously generated by a read operation.

WRITE_FIRST or Transparent Mode (Default)

In WRITE_FIRST mode, the input data is simultaneously written into memory and stored in the data output (transparent write), as shown in Figure 2. These waveforms correspond to latch mode when the optional output pipeline register is not used.

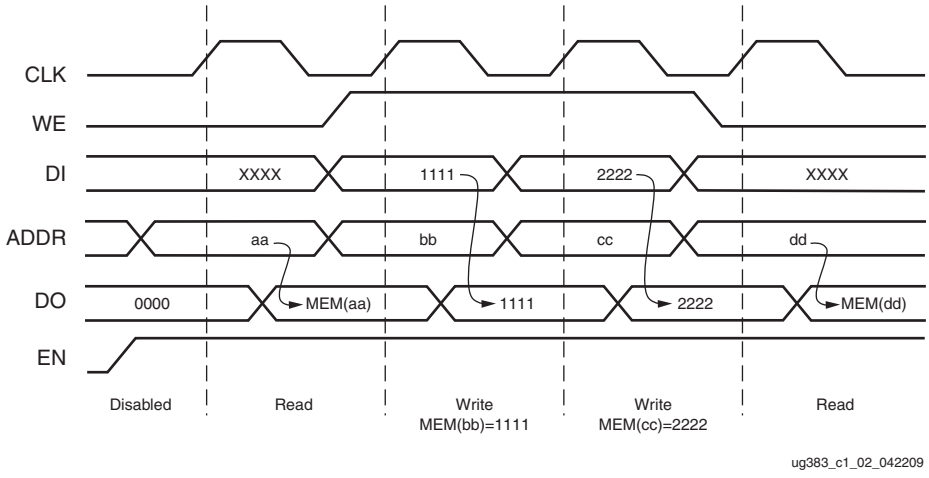

Figure 2: **WRITE_FIRST Mode Waveforms**

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 input data is being stored in memory (read before write). The waveforms in Figure 3 correspond to latch mode when the optional output pipeline register is not used.

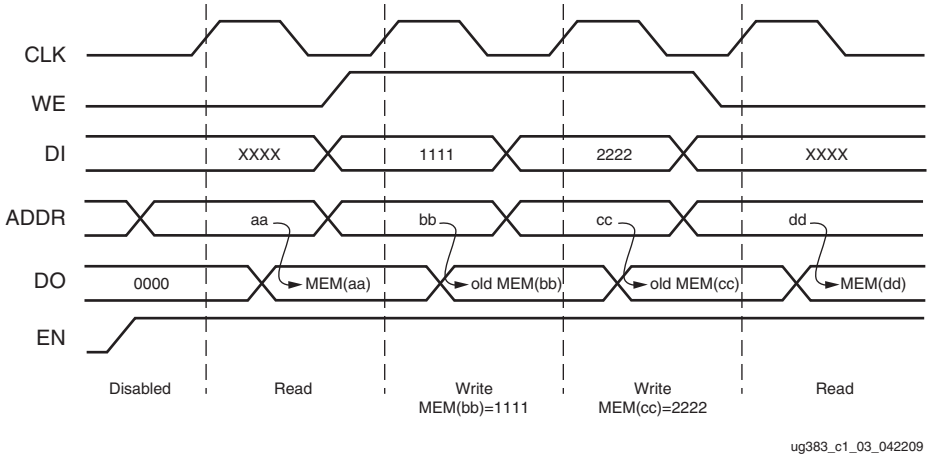

Figure 3: **READ_FIRST Mode Waveforms**

NO_CHANGE Mode

In NO_CHANGE mode, the output latches remain unchanged during a write operation. As shown in Figure 4, data output remains the last read data and is unaffected by a write operation on the same port. These waveforms correspond to latch mode when the optional output pipeline register is not used.

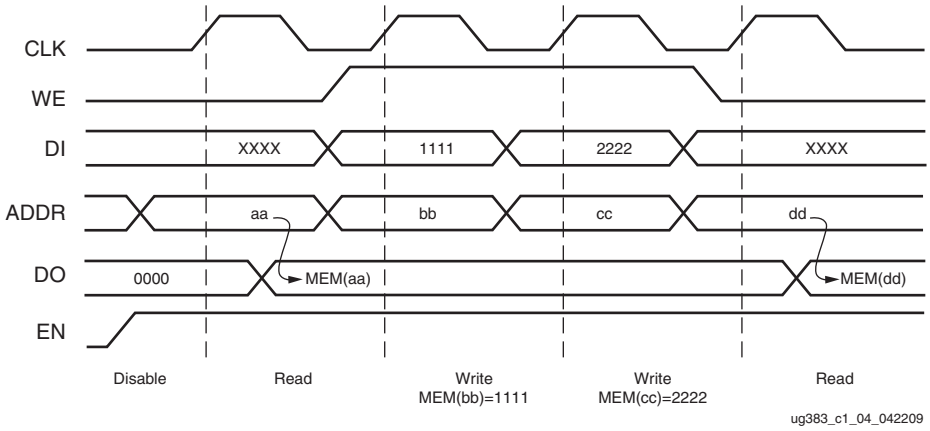

Figure 4: **NO_CHANGE Mode Waveforms**

Block RAM Operating Mode Inference Example

Note non-blocking assignments. Hence, the output databus contains the previous mem[address] contents.

Verilog

```
// 'write first' or transparent mode
always @(posedge clk) begin
  if(we) begin
    do <= data;
    mem[address] <= data;
  end else
    do <= mem[address];
end
```

```
// 'read first' or read before write mode(slower)
always @(posedge clk) begin
  if (we)
    mem[address] <= data;
  do <= mem[address];
end
```

```
// 'no change' mode
always @(posedge clk)
  if (we)
    mem[address] <= data;
  else
    do <= mem[address];
end
```

So which write mode is used here??

Verilog Code Example Inferring Single-Port Block SelectRAM

```
module ram_test(q, a, d, we, clk);

output [7:0] q;
input  [7:0] d;
input  [6:0] a;
input  clk, we;

reg [6:0] read_add;

/* The array of an array register ("mem") the RAM will be inferred from.
*/
reg [7:0] mem [127:0] /* synthesis syn_ramstyle = "block_ram */;

assign q = mem[read_add];

always @(posedge clk) begin
if(we)
/* Register RAM Data */
mem[a] <= d;
/* Register Read Address. Basic RAM support does not
require this address register.*/
read_add <= a;
end

endmodule
```

makes implementation in BRAM