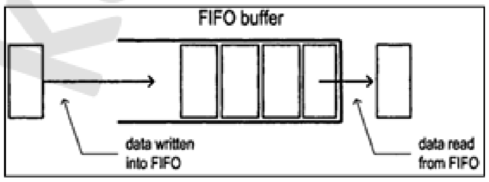
4) Implementing a FIFO (First-In-First-Out) memory. FIFO is memory structure that stores and retrieves data elements in the order they were added. The FIFO memory will be designed to have two main operations: writing (enqueuing) data and reading (dequeuing) data. We'll use two internal pointers (counters) to keep track of the write and read positions within the memory. The write pointer advances when new data is written, and the read pointer advances when data is read.



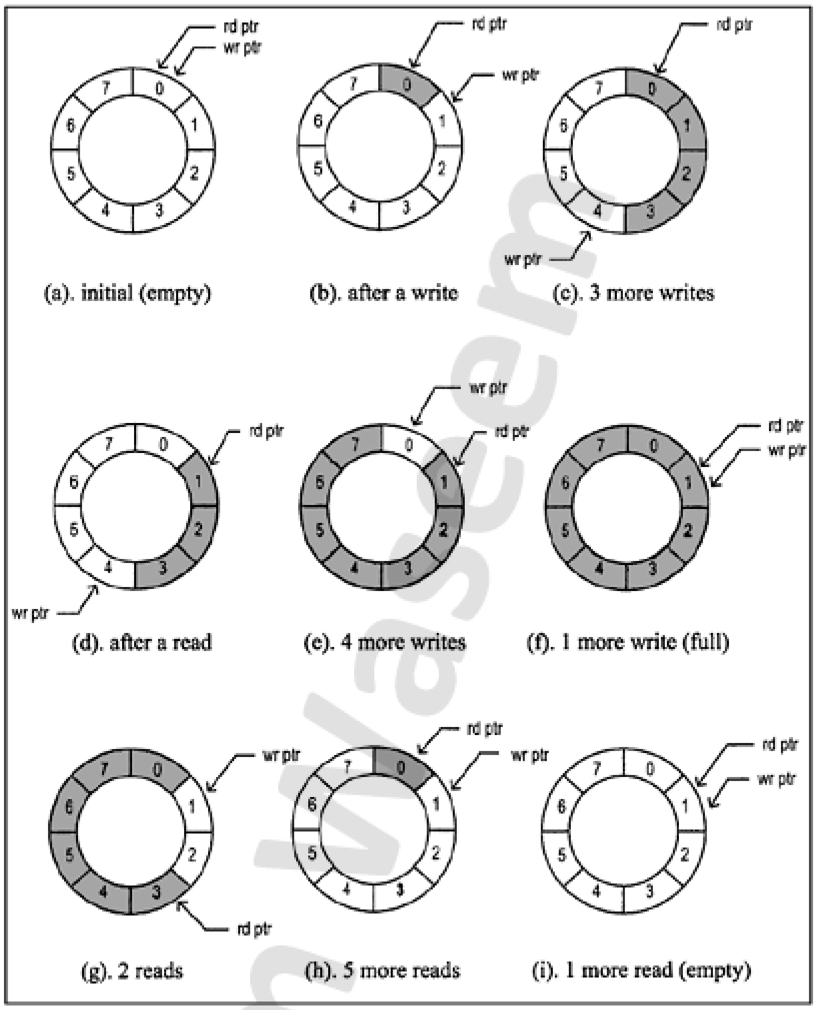


Figure 1 FIFO With depth of 8 words

Parameters

- FIFO_WIDTH: DATA in/out and memory word width (default: 16)
- FIFO_DEPTH: Memory depth (default: 512)

Ports

Port	Width	Direction	Function
din_a	FIFO_WIDTH	Input	Write Data: The input data bus used when writing the FIFO.
wen_a	1		Write Enable: If the FIFO is not full, asserting this signal causes data (on din_a) to be written into the FIFO
ren_b	1		Read Enable: If the FIFO is not empty, asserting this signal causes data (on dout_b) to be read from the FIFO
clk_a	1		Clock signal for port a, used in the writing operation
clk_b	1		Clock signal for port b, used in the reading operation
rst	1		Active high synchronous reset. It resets the dout_b, internal write counter & internal read counters
dout_b	FIFO_WIDTH	Output	Read Data: The output data bus used when reading from the FIFO.
full	1		Full Flag: When asserted, this signal indicates that the FIFO is full. Write requests are ignored when the FIFO is full, initiating a write when the FIFO is full is not destructive to the contents of the FIFO.
empty	1		Empty Flag: When asserted, this signal indicates that the FIFO is empty. Read requests are ignored when the FIFO is empty, initiating a read while empty is not destructive to the FIFO.