Parameterizable Synchronous Memory (Single-port read/write)

- A synch. single port memory with parameterized depth and width
- It supports a simple handshake :
  1. when valid is asserted the module performs either write (WR=1) or Read (WR=0) at provided addr on the same clk edge.
     → Drives ready=1 to indicate op^n was accepted, and for reads places the word on data_out.
  2. on reset, memory and outputs are cleared

- Parameters & ports :-
  1. depth, width, Addr_width = $clog2(depth) - sized memory and addr. width
  2. i/p : clk, rst, WR, data_in, valid, addr
  3. o/p : data_out (reg), ready (reg)

- Storage :-
  1. reg [width-1:0] memory [0:depth-1] the data array.
  2. int i, used for reset initiation

- Reset behaviours (1) data_out <= 0, ready <= 0
  (2) zero intialize entire memory in a loop.
- Normal op^n (on posedge clk when not rst)
  if valid (1) ready <= 1, (acknowledge the request this cycle)
  (2) WR = 1, memory[addr] <= data_in (synchronous write)
  (3) WR = 0, data_out <= memory[addr] (synch Read - data_out updated)
  else valid (deasserted)
  ready <= 0
- Behavioural note
  (1) Single cycle synch read and write triggered by valid. Both op^n are acknowledged immediatly via ready the same cycle.
  (2) for read, data_out is updated synchronously (external logic must sample data_out after the clock edge where valid && !WR)
  (3) No write/read arbitration beyond valid If valid toggles per clk you get one op^n per cycle.
  (4) Memory initialized on rst - good for deterministic simulation.