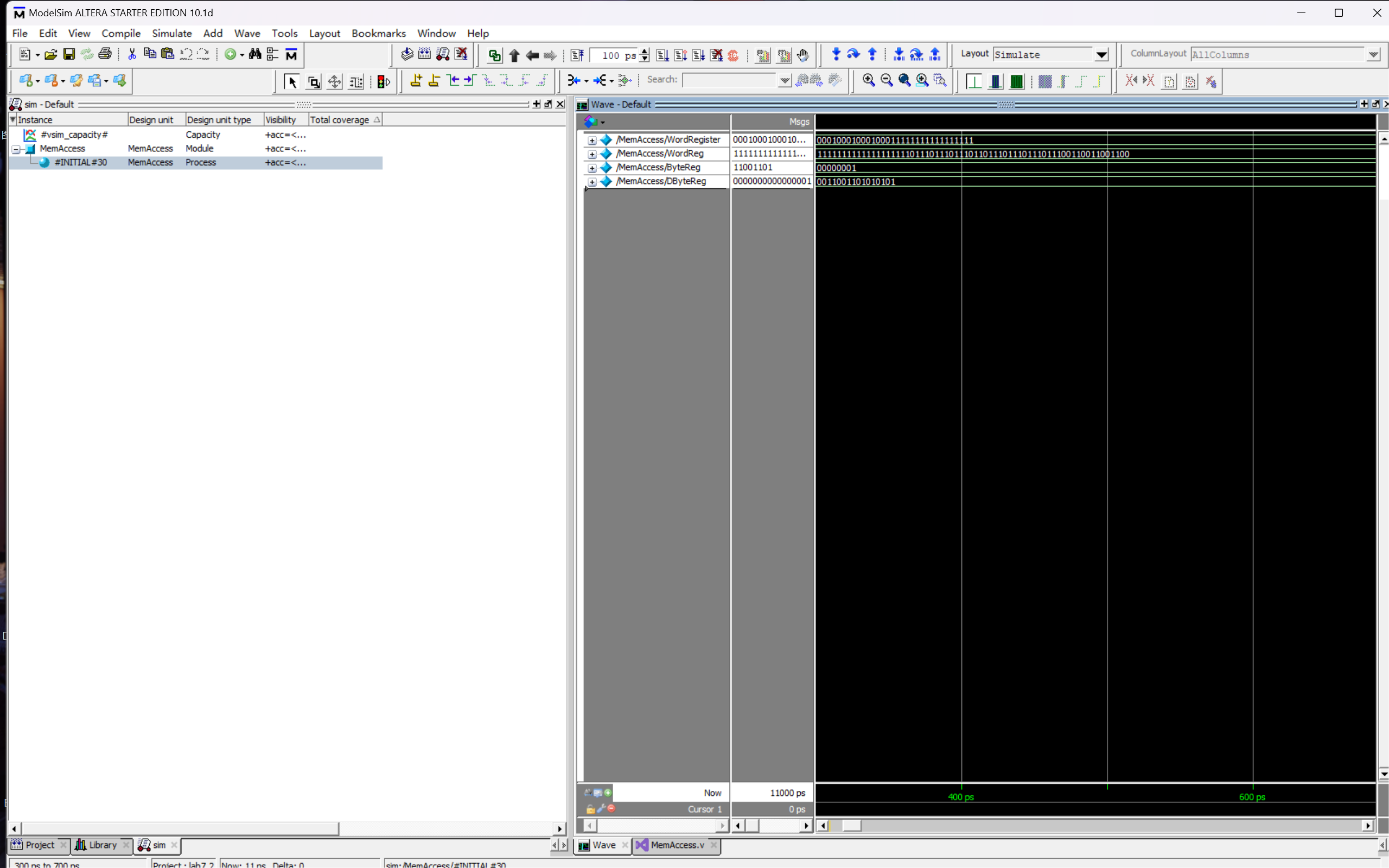
Step1：



Step2：

parameter AdrHi = 4: 这是一个模块参数，用于设置地址总线的宽度。在这里，默认设置为4位，意味着地址总线的范围为0~15。

output Dready, ParityErr, output[31:0] DataO, input[31:0] DataI, input[AdrHi:0] Addr, input ClockIn, ChipEna, Read, Write: 这是模块的输入输出端口。

localparam MemHi = (1<<AdrHi+1) - 1;: 这是一个本地参数，用于计算存储器数组的最大地址。在这里，它计算为2^5 - 1，即31，因为AdrHi设置为4。

reg[32:0] Storage[MemHi:0];: 这是一个32位宽的存储器数组Storage，大小为32x32，其中每个存储单元是32位宽的。

reg[31:0] DataOreg;: 这是一个寄存器，用于暂存从存储器中读取的数据。

wire ChipClock;: 这是一个用于存储经过芯片使能ChipEna控制的时钟信号。

reg ParityReg, DreadyReg;: 这是用于存储输出端口ParityErr和Dready的寄存器。

assign #1 ChipClock = (ChipEna==1'b1)? ClockIn : 1'b0;: 这是一个组合逻辑赋值语句，用于根据ChipEna信号来控制时钟信号。当ChipEna为1时，ChipClock等于ClockIn，否则等于0。

assign #3 Dready = (ChipEna==1'b1)? DreadyReg : 1'b0;: 这是一个组合逻辑赋值语句，用于控制输出端口Dready。当ChipEna为1时，Dready等于DreadyReg，否则等于0。

assign #2 DataO = (ChipEna==1'b1)? DataOreg : 'bz;: 这是一个组合逻辑赋值语句，用于控制输出端口DataO。当ChipEna为1时，DataO等于DataOreg，否则等于高阻态。

always@(posedge ChipClock): 这是一个时钟边沿触发的always块，它在ChipClock上升沿触发时执行。

if (ChipEna==1'b1): 如果ChipEna为1，则执行以下语句。

if (Read==1'b1): 如果Read为1，则表示要从存储器中读取数据。

ParityReg <= (^(Storage[Addr])==1'b0)? 1'b0 : 1'b1;: 这是计算并存储读取数据的奇偶校验位。

DataOreg <= Storage[Addr];: 将存储器中指定地址的数据暂存到DataOreg中。

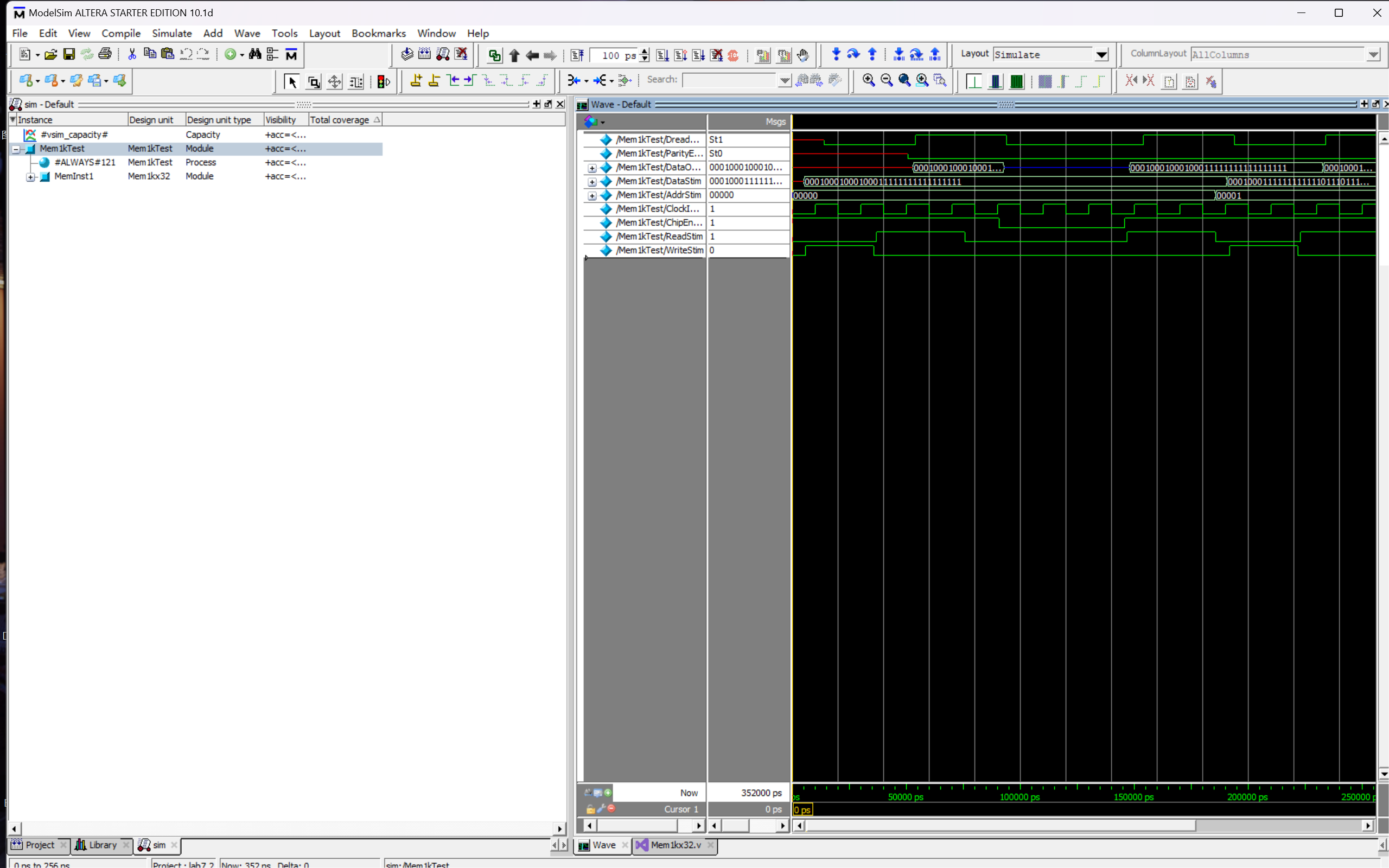
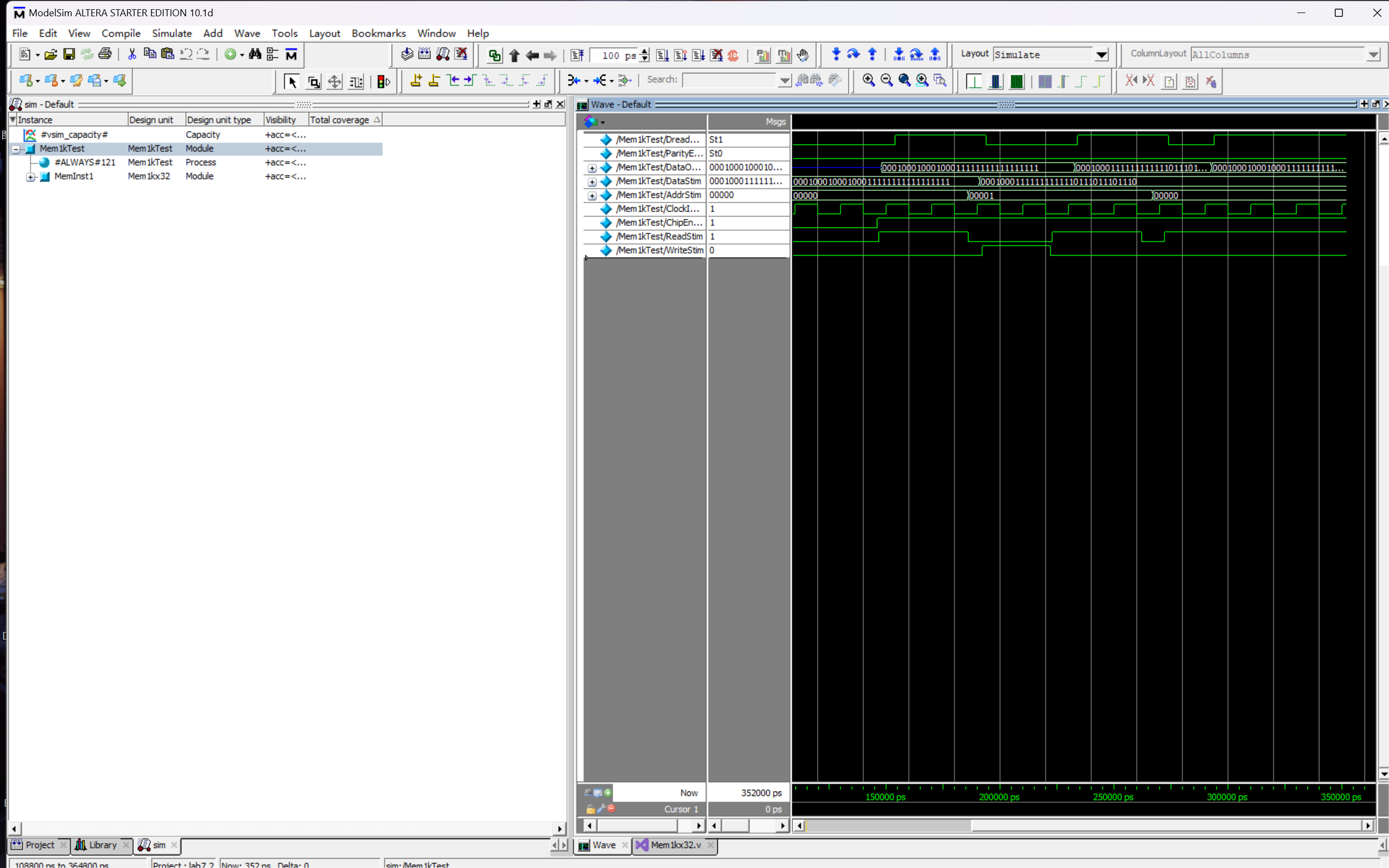
DreadyReg <= 1'b1;: 设置输出端口Dready为1，表示数据有效。

if (Write==1'b1): 如果Write为1，则表示要向存储器写入数据。

DreadyReg <= 1'b0;: 设置输出端口Dready为0，表示数据无效。

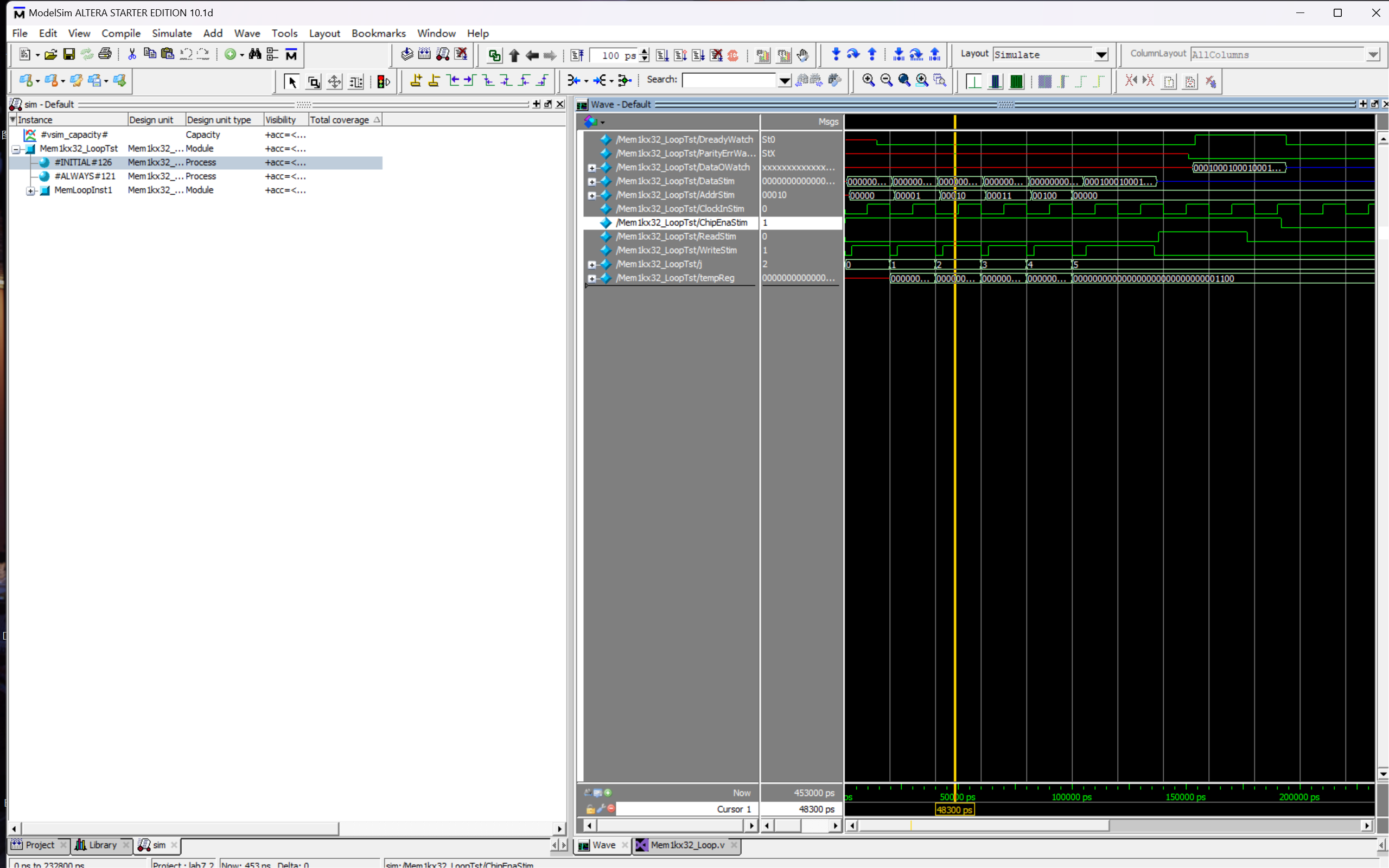
Storage[Addr] <= {^DataI[31:0], DataI[31:0]};: 将输入端口DataI的数据写入到存储器指定的地址，并同时计算并存储写入数据的奇偶校验位。

else: 如果既不是读取也不是写入，则说明不进行操作，设置输出端口Dready为0。



只有在时钟上升沿检测到读或写的控制信号变化，读写才会起作用。若都无效，则读数据端口由上一次读出的值驱动，若都有效，则进行读操作，但数据无效。

Step3，4：



Step5：

