18.FIFO IP 核使用实验例程

18.1实验简介

FIFO 即先入先出，在 FPGA 中，FIFO 的作用就是对存储进来的数据具有一个先入先出特性的一个缓存器，经常用作数据缓存或者进行数据跨时钟域传输。FIFO 和 RAM 最大的区别就是 FIFO 不需要地址，采用的是顺序写入，顺序读出。

在紫光的 IP 工具中又分为 Distribute FIFO 和 DRM FIFO，其实就是用不同的资源去构成，前者 Distribute FIFO 也就是分布式 FIFO，使用的是片上的 LUT 资源去构成，而 DRM FIFO 使用的是片上的 DRM 资源去构成，DRM 构成的 FIFO 其性能大于 LUT 资源构成的，不仅容量更大，且可配置更多功能。

本实验介绍 DRM Based FIFO，想要了解 Distribute FIFO 请参考紫光的 IP 手册

18.2实验目的

掌握 FIFO 的写入和读出并灵活根据需求设计 FIFO。

18.3实验设计

设计一个读写交替的状态机：当写入 128 个数据后开始读取，读出所有写入数据之后开始第二次写。这样交替进行读和写的操作

IP 配置

进入 IP 选择界面后

第一步选择 DRM FIFO；

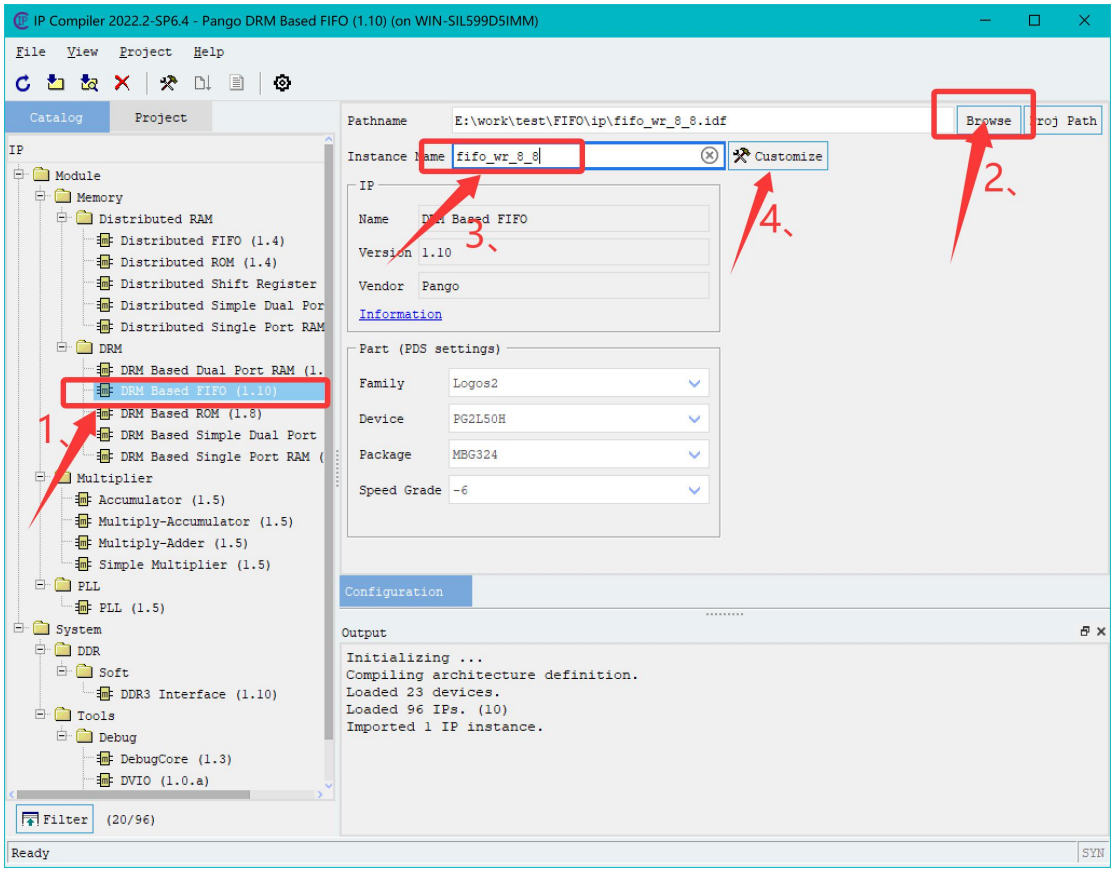
第二步选择我们 IP 的存放路径；

第三步编辑 IP 的名称，这里我们名称为 fifo_wr_8_8。从名称可以获取的信息为这个 IP 和的类型，他的功能是写入用户数据，写位宽为 8bit，读位宽为 8bit。

命名规则推荐：IP 类型_IP 功能_写入位宽_读出位宽；

这样命名可以直观的看到 IP 的重要配置信息，我们用户一般关心的参数一目了然

第四步进入 IP 配置页进行参数的配置



进入参数配置界面

上面的 DRM Resource Usage 可选择 DRM 资源模式及统计 DRM 资源使用数量，此处保持默认；

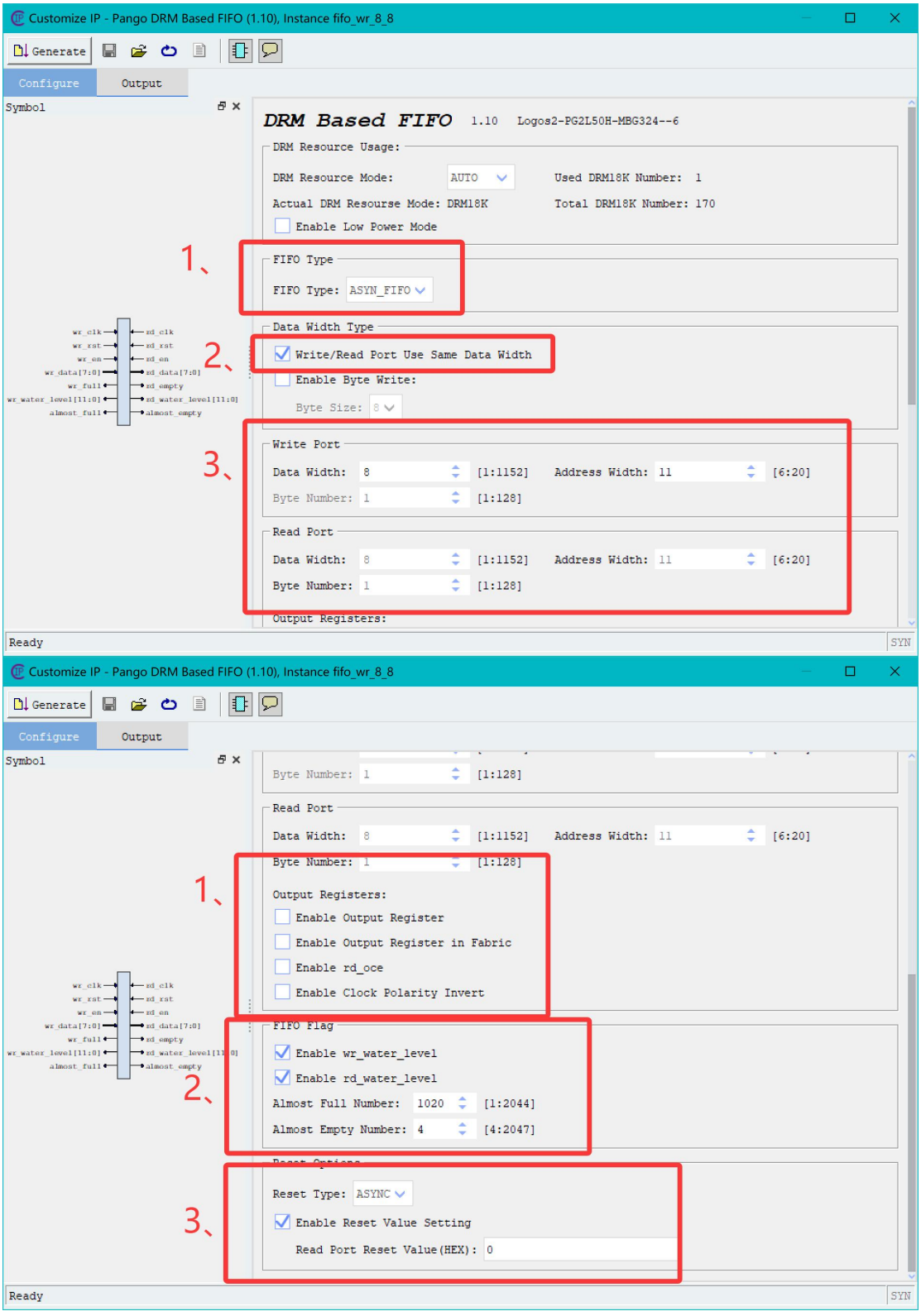
FIFO Type 我们选择异步 FIFO。对于 FIFO 的类型有两种，一种是同步 FIFO，读写端口的时钟和复位是共用的。另一种是异步 FIFO，读写端口时钟和复位独立控制，当然我们也可以给相同的时钟和复位，我们本次实验就是这么做的。我们一般使用的时候都会选用异步 FIFO，这样方便数据的跨时钟处理。

接下来就是配置我们的数据位宽以及 FIFO 深度了。因为我们勾选了读写端口位宽一致的选项，因此只需要配置读端口位宽及深度即可。一定要注意的是我们 FIFO 的深度选择。合适即可，不可过大浪费资源，也不可过小不够使用。我们不启用字节读写功能，想要详细了解 IP 每个参数的作用请参考 IP 手册，内部详细介绍了每个参数的作用。

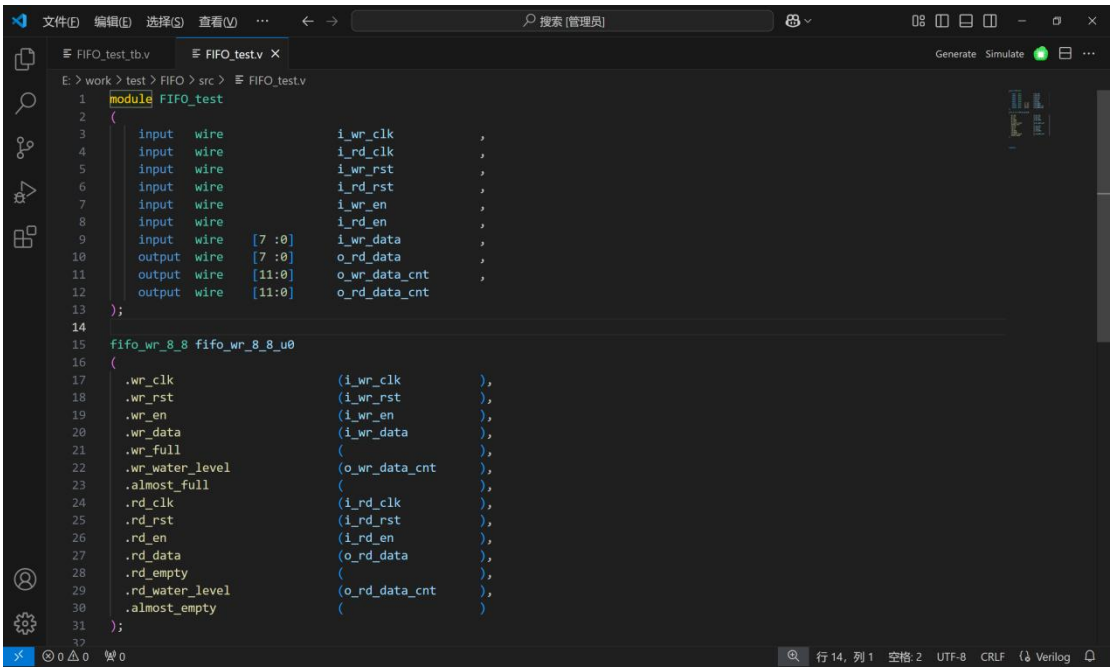
Output registers 是代表我们输出是否需要用寄存器打一拍，我们默认不勾选。

我们勾选上 Enable wr_water level 和 Enable rd_water level。这两个信号的含义就是代表我们当前 FIFO 中数据量的多少。通常根据这两个信号可以控制我们的读使能和写使能，在本次实验中我们就是这样做的。

还有两个 Almost 信号是表示我们的 FIFO 将要满/空，可以设置什么时候触发，比如如图设置的 1020 即代表当写入 1020 个数据后这个 Almosti Full 信号就会拉高，同理剩 4 个数据的时候 Almost Empty 信号就会拉高。



所有参数配置完成之后我们点击 Generate，
例化我们的 IP 核
测试数据我们在仿真文件中编写



仿真文件源码如下

写状态机：

当写入数据小于 128 时我们拉高写使能，直到 FIFO 内部写数据达到 127 之后，下一个时钟周期拉低写使能，总共拉高 128 个时钟周期，写入 128 个递增数据

```
always@(posedge r_clk_100M) begin
      if(r_rst_100M)
      begin
            r_wr_state    <=   1'd0;
            r_wr_en       <=   1'd0;
            r_wr_data     <=   8'd0;
      end
      else
      begin
            case(r_wr_state)
                  P_WR_DATA:begin
                        if(w_wr_data_cnt == 127)begin
                                    r_wr_en      <= 1'd0;
                                    r_wr_data    <= 8'd0;
                                    r_wr_state <= 1'd1;
                        end else begin
                                    r_wr_en      <= 1'd1;
                                    r_wr_data    <= r_wr_data+1'b1;
                                    r_wr_state <= 1'd0;
                        end
                  end
                  P_WR_WAIT:begin
                        if(r_rd_cnt == 127)
```

```
                              r_wr_state <= 1'd0;
                    end
                    default:r_wr_state <=1'd0;
              endcase
        end
end
```

读状态机：

当 FIFO 中读端口的数据不少于 128 个时拉高读使能，并开始计数，直到计满 128 个数据后拉低读使能

```
always@(posedge r_clk_100M) begin
      if(r_rst_100M)
      begin
            r_rd_state   <=   1'd0;
            r_rd_en       <=   1'd0;
            r_rd_cnt      <=   8'd0;
      end
      else
      begin
            case(r_rd_state)
                  P_RD_DATA: begin
                        if(w_rd_data_cnt >= 8'd128) begin
                              r_rd_state   <= 1'd1;
                              r_rd_en       <= 1'd1;
                        end else begin
                              r_rd_cnt      <=8'd0;
                              r_rd_state   <=1'd0;
                        end
                  end
                  P_RD_WAIT:   begin
                        r_rd_cnt      <= r_rd_cnt + 1'b1;
                        if(r_rd_cnt == 127) begin
                              r_rd_en       <= 1'd0;
                              r_rd_state   <= 1'd0;
                        end
                  end
                  default: r_rd_state <=   1'd0;
            endcase
      end
end
```

需要注意要例化这一语句不然无法进入仿真

```
    GTP_GRS GRS_INST
    (
```

```
        .GRS_N(1'b1)
    );
```

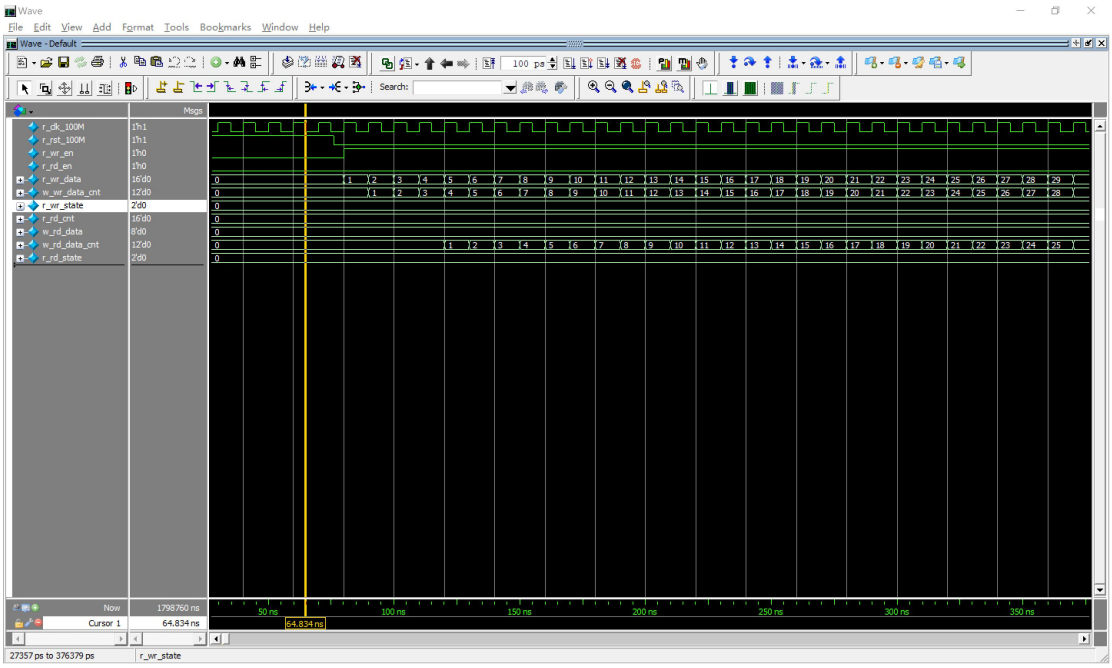
端口连接

```
FIFO_test FIFO_test_u0
(
    .i_wr_clk                   (r_clk_100M         ),
    .i_rd_clk                   (r_clk_100M         ),
    .i_wr_rst                   (r_rst_100M         ),
    .i_rd_rst                   (r_rst_100M         ),
    .i_wr_en                    (r_wr_en            ),
    .i_rd_en                    (r_rd_en            ),
    .i_wr_data                  (r_wr_data          )
    .o_rd_data                  (w_rd_data          )
    .o_wr_data_cnt              (w_wr_data_cnt      ),
    .o_rd_data_cnt              (w_rd_data_cnt      )
);
```

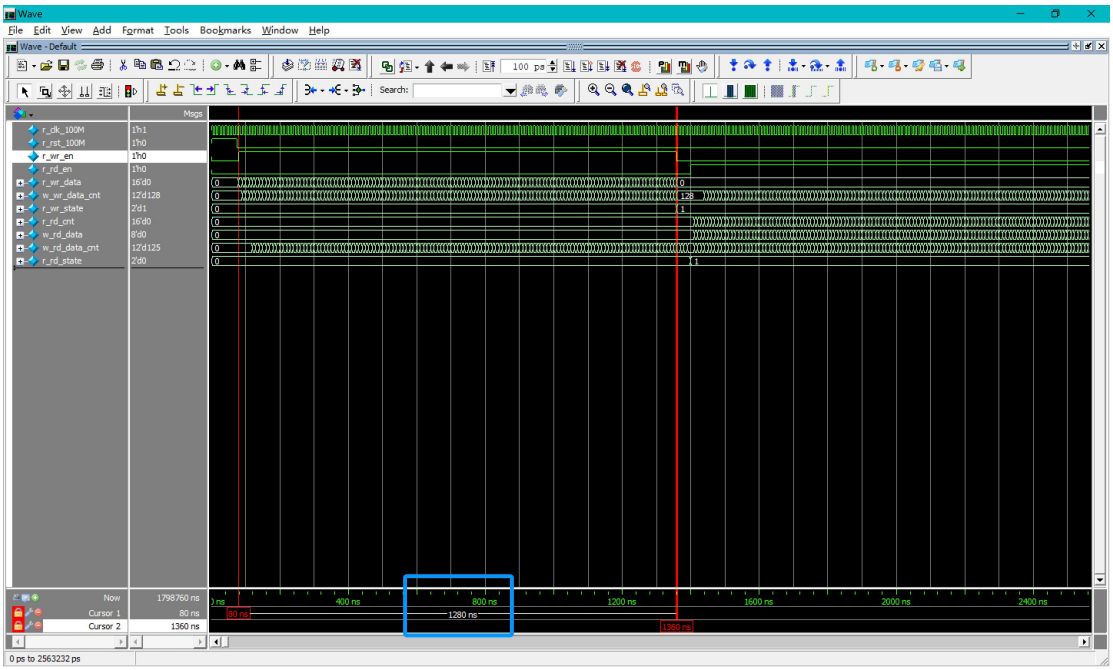
18.4仿真波形

可以看到和我们的设计一致：复位完成后拉高写使能，写入递增数据

从下图也可以看到我们写使能拉高了 128 个时钟周期（1280ns = 128 cycle * 10ns)

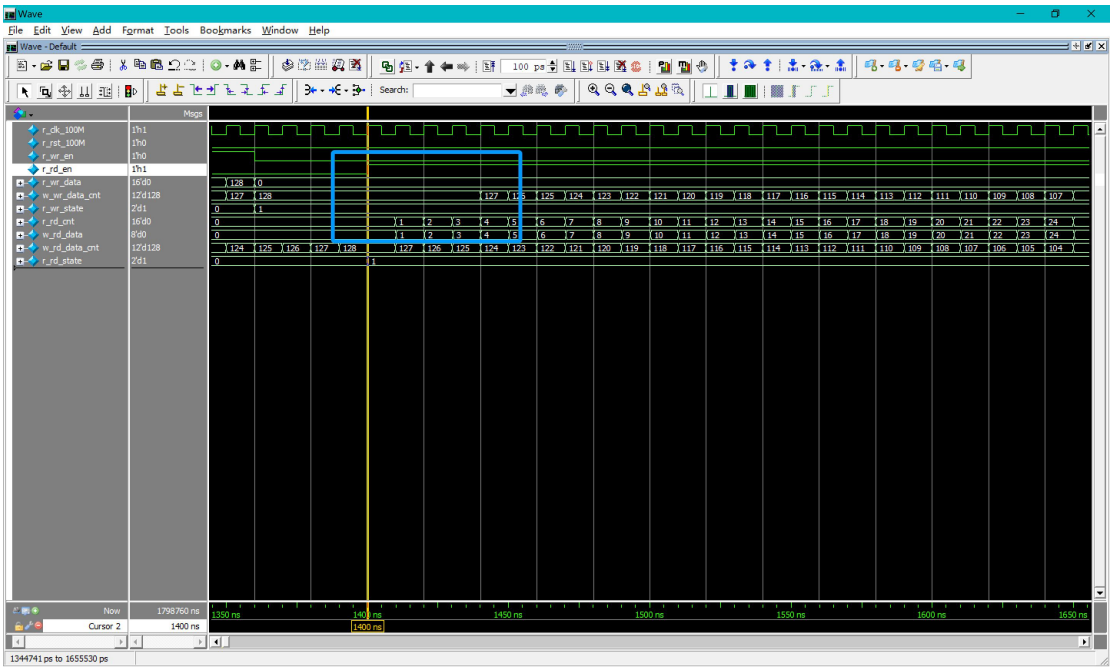


写使能拉高

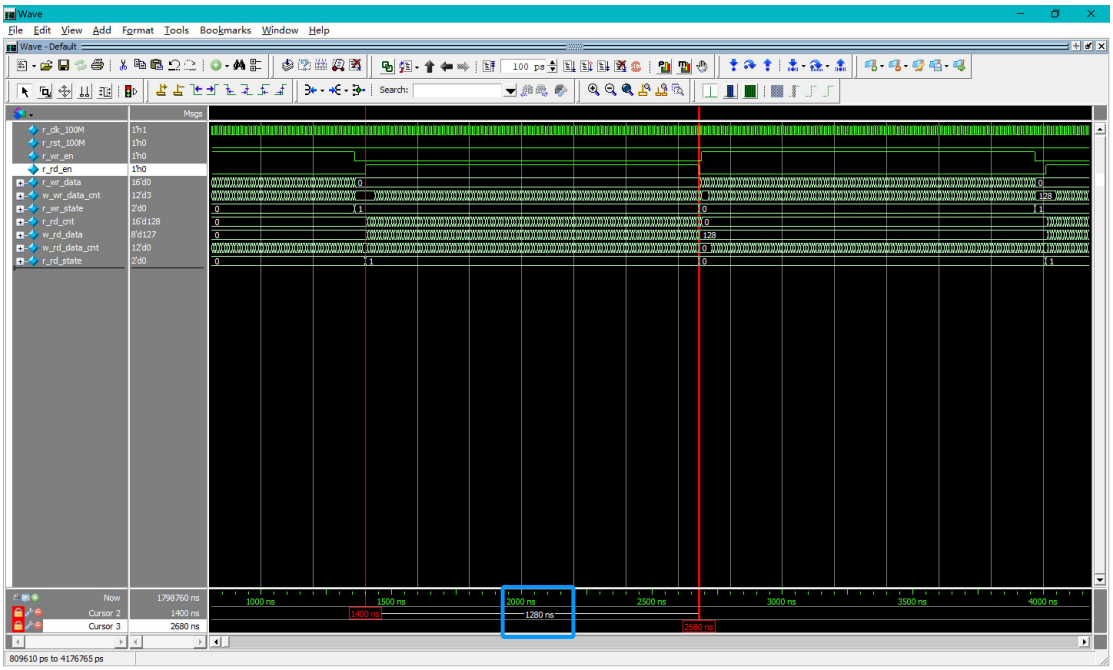


写使能周期

读端口跟我们的设计也是一致，读出数据也正确
并且读使能同样拉高了 128 个周期



读使能拉高



读使能周期

至此，我们 FIFO 的实验已经成功完成！！！