UART\_SRAM使用说明

UART\_SRAM模块实现了通过UART接口向EGO1片上的SRAM中写入数据的功能。在一个硬件项目的测试中，可能需要更换不同的testbench数据，本模块通过SRAM来进行数据的写入，避免了修改代码之后多次重新综合、布线与烧写FPGA的过程。

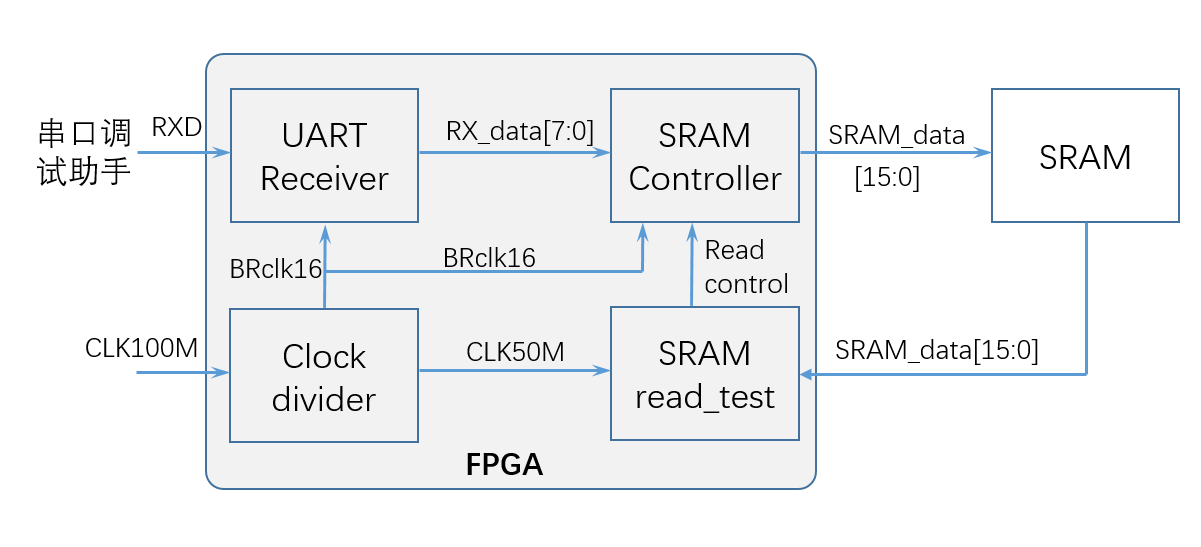


图1 UART\_SRAM模块结果图

图1给出了UART\_SRAM模块的整体结构图。

**UART receiver模块：**测试数据从串口调试助手输入，通过RXD端口进入FPGA。采用100MHz系统时钟，分频得到9600波特率的16倍采样时钟，接受UART传入的数据，并将得到的8bit RX\_data传给SRAM controller。

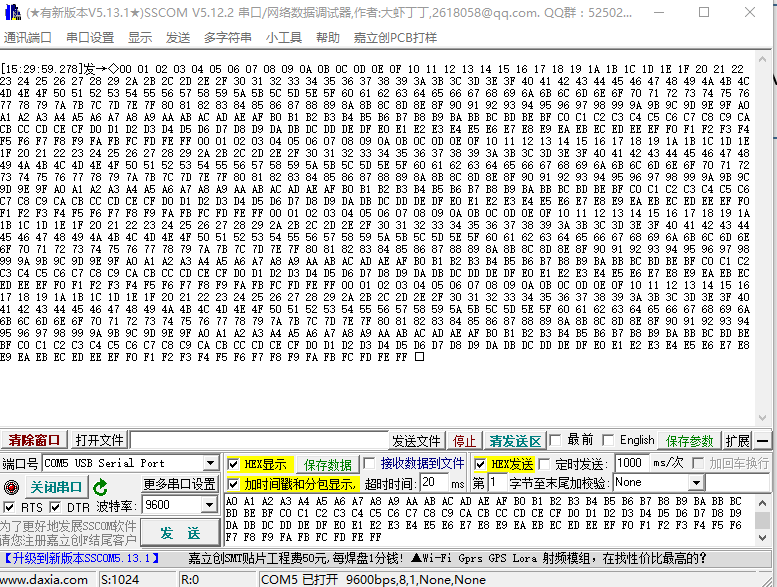


图2 串口调试助手输入数据示意图

SRAM controller模块：负责选择向SRAM写入数据或从SRAM读出数据。写入数据部分，由于SRAM数据位宽为16bit，因此每两次的8bit RX\_data数据会拼接成一个16bit数据写入SRAM，写入的地址从0开始依次增加。读出数据部分，接受来自SRAM read\_test模块的控制信号（地址，使能信号），从SRAM读出数据。

SRAM read\_test模块：用于测试写入并读出的数据是否正确，测试采用了连续输入4次00-FF共1024个8bit数据（如图2），写入到SRAM地址0-511的位置。读出时，每两次读出的16bit数据合成一个32bit数据（如hex03020100），与标准结果对比，输出compare\_result.

Clock divider模块：生成9600波特率16倍采样时钟和50MHz时钟，注意SRAM读写时钟最好不要高于50MHz，否则可能不满足SRAM时序要求而出错。也可以根据SRAM的datasheet自行分析SRAM读写允许的最高时钟频率。

SRAM模块：SRAM是在EGO片上已有的异步SRAM芯片，大小为128Kx16bit，地址位[18:0]，数据位宽[15:0]。接口可以参考user guide和本项目中的xdc文件，更详细的可以看IS61WV12816BLL SRAM芯片的datasheet。

测试方式：

1. Reset操作，低电平有效，接口为SW0-R1，如图3所示，此时左侧8个LED中LED1（SRAM写使能，低电平有效）和LED3（compare result，1为正确）会亮。

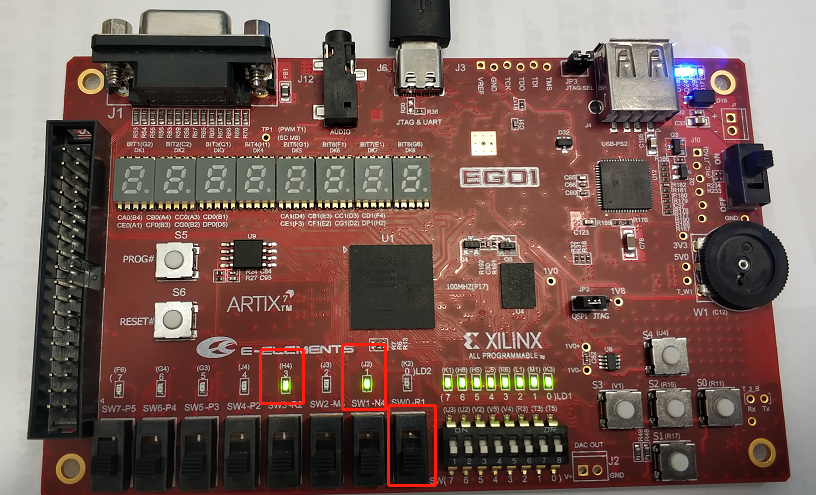


图3 reset操作下EGO1示意图

1. Reset置高电平，通过串口调试助手输入图2所示数据（重复4次的00-FF），如图4所示，此时左侧8个LED中的LED0-4会亮。LED0是完成SRAM读取操作的信号，LED2是完成UART数据传输的信号。此时LED3仍为1，说明0-511地址的SRAM数据和预想一致。

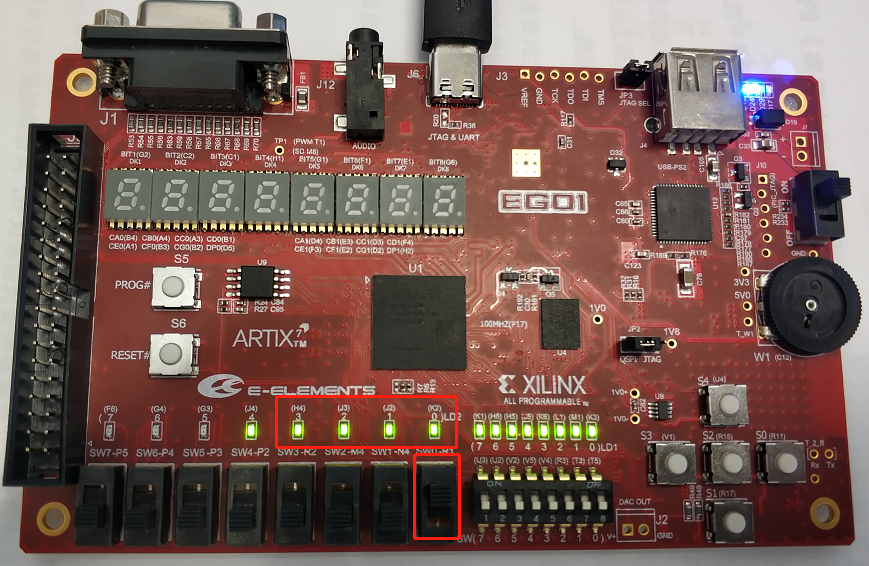


图4 运行情况下EGO1示意图

1. 测试模式下，可以通过右侧8个LED小灯观察SRAM中的数据（高8bit或低8bit），下方的8个小SW表示地址的[7:0]位，高位[18:8]设置为0（参考SRAM controller代码）。左侧的SW1-N4是开启测试模式的开关，高电平有效。SW2-M4为控制显示数据高8bit或低8bit的信号，高电平显示高8bit，低电平显示低8bit。左侧LED4-7为SRAM地址的[3:0]位。如图5所示，此时显示地址8的低8bit，结果为16.

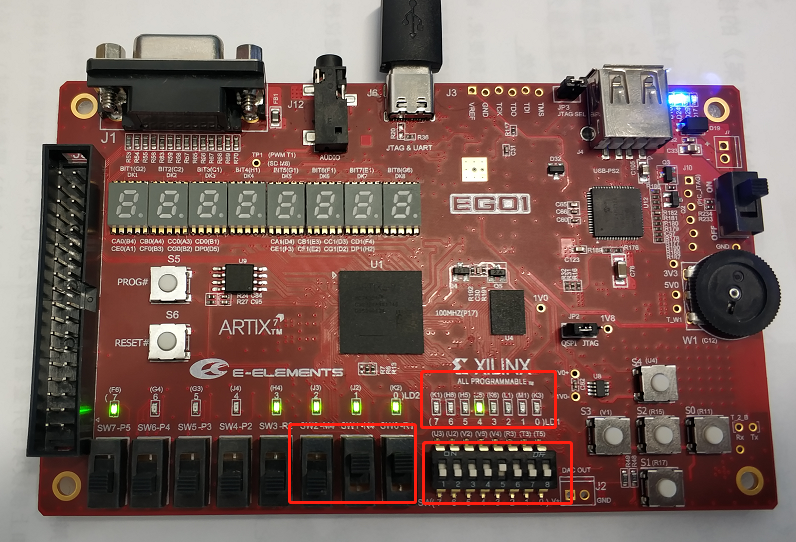


图5 测试模式下EGO1示意图

修改注意事项：

1. 可以对本模块进行修改实现对于一个硬件项目测试不同testbench数据的功能，避免多次vivado的综合与烧写操作。
2. 目前代码中每次允许从UART写入的数据为1024个8bit数据，可以修改UART\_sim.v中第10行的参数进行修改。建议设置一个较大的值，然后每次的testbench如果长度不同，可以使用无效数据补齐数量。或者修改加入终止符号等等。
3. 由于从SRAM读取数据使用50MHz时钟，且SRAM位宽为16bit，所以如果希望得到32bit的CPU指令数据，需要使用25MHz的时钟运行CPU。（因为每两个读操作才能得到一个32bit数据）
4. 注意SRAM\_data是一个inout端口，既是输入也是输出，因此使用了三态门设计，注意在sram\_controller.v中第28行的写法，如果需要修改，请避免把该信号改为一个恒定的输入或输出信号，否则可能出错。

文件说明：

1. UART\_SRAM使用说明.docx
2. IS61WV12816BLL datasheet
3. UART\_SRAM源代码文件夹
4. UART\_SRAM工程文件夹