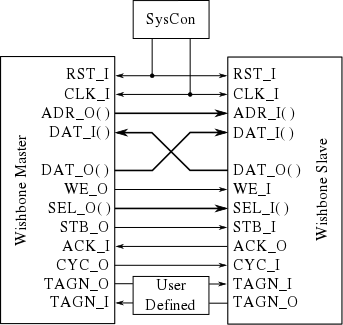
Wishbone总线

Wishbone 总线最先是由Silicore公司提出，现在己被移交给OpenCores组织维护，它通过在IP核之间建立一个通用接口完成互连。可以用于在软核、固核以及硬核之间进行互联。在NEHCPU项目中，我们使用了Wishbone总线来进行CPU与外设的交互，以及对结构冲突进行处理。

Wishbone的大致结构如下：



考虑到我们的读写都是一次读取一个地址，所以我们只使用了如下的接口：

Slave端：

RST\_I：重置信号

CLK\_I：时钟信号

ADR\_I：Wishbone总线地址

DAT\_I：从Master发送的数据

DAT\_O：向Master发送的数据

WE\_I：写使能

SEL\_I：是一个四位的信号，代表数据的哪几位有效  
STB\_I和CYC\_I：选通信号，当二者都为高电平时才可以进行通信

ACK\_O：确认信号

Master端：

RST\_I：重置信号

CLK\_I：时钟信号

ADR\_O：Wishbone总线地址

DAT\_I：从Slave发送的数据

DAT\_O：向Slave发送的数据

WE\_O：写使能

SEL\_O：是一个四位的信号，代表数据的哪几位有效  
STB\_O和CYC\_O：选通信号，当二者都为高电平时才可以进行通信

ACK\_I：确认信号

以此即可实现Wishbone总线来实现CPU对SRAM和外设的读写操作。有关Wishbone总线的选择，我们使用了WB\_Conmax这个开源实现的版本。这个版本支持8个主设备和16个从设备，且支持优先级配置，允许多对主从设备同时通信。默认，地址高四位为从设备的索引号，从0~15依次代表与0~15号从设备进行通信。在NEHCPU中，我们有四个从设备，分别为SRAM、FLASH、存放bootloader的ROM、串口和VGA。所以在NEHCPU我们约定Wishbone总线的地址范围如下：

|  |  |  |
| --- | --- | --- |
| 从设备 | Wishbone总线地址范围 | 序号 |
| SRAM | 0x00000000~0x0FFFFFFF | 0 |
| FLASH | 0x10000000~0x1FFFFFFF | 1 |
| ROM | 0x20000000~0x2FFFFFFF | 2 |
| UART | 0x30000000~0x3FFFFFFF | 3 |
| VGA | 0x40000000~0x4FFFFFFF | 4 |

外设

NEHCPU通过wishbone总线连接五个从设备。五个从设备的信息如下：

ROM

ROM的作用是存储bootloader。在bootloader中不仅能够将操作系统从FLASH加载到SRAM中，还可以验证FLASH和SRAM的时序是否正常。如果不正常会卡死在bootloader中。ROM的物理地址为0xBFC00000~0xBFC00FFF，转化为虚拟地址即为0x1FC00000~0X1FC00FFF。

ROM的速度是非常快的，所以在返回数据的时候不需要等待若干个周期，直接依照wishbone总线的时序规范返回即可。

SRAM

SRAM的物理地址为0x00000000~0x007FFFFF，由于实际上SRAM有两片，所以在NEHCPU中将有效地址的最高位（23位）设置为Select位，通过这一位来选择正确的SRAM。

SRAM的读写需要一定的时间，在NEHCPU中共使用了3个50M时钟周期来等待SRAM返回数据。NEHCPU使用一个小型的状态自动机来实现等待的操作。

FLASH

Flash的物理地址为0xBE000000~0xBEFFFFFF，由于Flash的字长为16bit，所以读出来的数据只有低16位有效。Flash读取十分缓慢，所以在NEHCPU中使用了15个50M时钟周期来等待Flash返回结果。

串口

NEHCPU使用了<http://www.fpga4fun.com/SerialInterface.html>的开源代码来直接与串口进行交互，并自行实现了一个其与wishbone总线交互的接口。为了更方便地与串口进行交互，我们自行开发了一个基于QT的串口小工具来代替串口调试助手。这个工具可以很有效地模拟“终端”，且可以跨平台编译使用。

VGA

对于VGA的处理分为两部分。

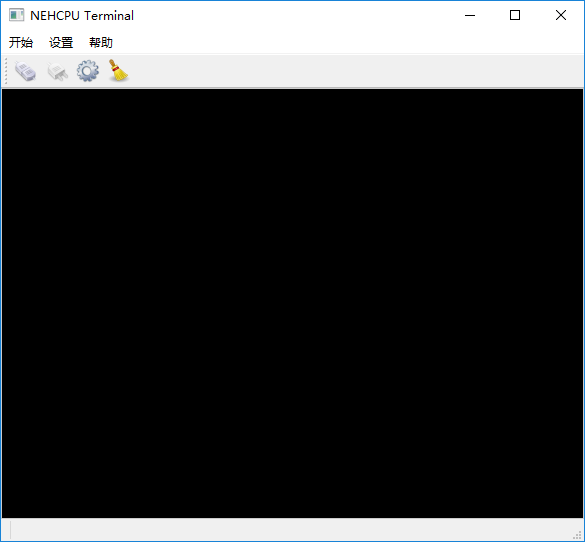
首先操作系统应该可以通过对某些地址的写操作来向VGA输出东西，为此我们在FPGA内部实现了一个大小为0x96000的“显存”，且实现了一个其与wishbone总线交互的接口。

接下来是显示部分。显示的格式为800\*600（75Hz），但是经过我们的测试，如果存储640000个像素，会大大拖慢操作系统的运行速度。于是我们这里使用了400\*300扩展到800\*600显示的方案。同时为了加速寻址，实际存储的大小为512\*300，且只有400\*300有效。这样一个异步执行的VGA驱动程序就可以不断地扫描显存并输出。

考虑到“终端”会有大量滚屏情况的出现，这里还采用了一个优化，就是在每输出一行后，驱动程序输出VGA的首地址会往后移一行。操作系统内部也做一样的处理，就可以实现每输出一行只需要向VGA的显存输出一行的数据。这样大幅加快了显示速度。

VGA显存的物理地址为[0xBA000000,0xBA096000]，而VGA首地址的物理地址为0xBA096000。

终端工具简介



由于串口调试工具不能有效地实现“终端”这一功能，我们开发了一个终端工具。使用这个工具可以轻松查看串口的信息，连接指定的串口，接受串口信息并向串口发送信息。

这个工具是基于QT开发的，最低的QT版本为QT5.5。使用QT5.5以上的版本直接编译运行即可。

源代码已放在压缩包中。