数字信号分析仪参考设计

使用FPGA电路逻辑与NIOSII处理器软核

用绳量给我的地界

坐落在佳美之处

我的产业实在美好

杜伟韬

duweitao@cuc.edu.cn

2012年 8月 25日于 北京定福庄

目 录

0 引言

0.1 关于本参考设计

本设计试图通过一个例子来展示，数字电路设计、C语言编程、单片机开发、数字信号处理算法设计，这些本科生课程中所学的知识与技能在一个电子系统中协同工作的案例。该设计可以在采用Terasic出品Altera公司FPGA芯片开发板上运行，支持的开发板型号为DE0、DE2-115，需要在开发板外接VGA显示器。

0.2 为什么使用软核处理器

在某些电子技术论坛里关于软核处理器的使用存在着各种观点。总结来说，基于FPGA的软核处理器有以下的不足和优势。首先列举出不足之处：

- 运行速度较慢，通常的运行主频在100MHz左右
- 对于32位的处理器软核，通常会消耗较多的FPGA片内RAM块
- 在系统中添加处理器软核后，电路编译的时间会增长。
- 通常要外挂存储器，增加电路板的物料成本

另一方面，软核处理器也存在着以下的先天优势：

- 软核处理器是最原生的、便于和FPGA中的逻辑电路整合的处理器
- 具有高度的灵活性，可以定制高性能的专用指令（浮点、除法）
- 同样，可以牺牲处理性能从而得到电路面积的高度优化。
- 处理器软核之外的电路逻辑可以非常方便的与之进行耦合
- 可以方便的用FPGA内嵌的逻辑分析仪进行处理器的总线动作分析。
- 如果有必要，可以放置多个软核处理器增加计算性能。

至于在某个采用FPGA芯片的项目方案制定过程中，可能会存在多种因素影响设计者决定是否采用处理器软核，比如制造成本、开发周期、处理器性能、工具链等等，这些因素分析起来很复杂，此处就不过多讨论了。

本参考设计采用FPGA软核的原因很简单，这是最便于展示如何使用处理器和其外围数字逻辑进行系统设计的选择。

0.3 请了解一下计算机体系结构

当前的嵌入式开发领域中，32位的处理器成为主流，对于先修过8位单片机及其汇编语言编程的电子类专业的本科生而言，如果后续要学习32位嵌入式处理器的开发，推荐在阅读相关厂商手册的同时，也阅读一些计算机体系结构的教科书，这对于快速的学习和掌握不同的处理器很有帮助。尽管嵌入式处理器的型

号众多，并且在指令集、存储模式、外设控制器等方面也各有不同，但是它们均是按照计算机体系结构所定义出的框架来进行设计的。

目前的嵌入式处理器几乎无一例外的支持C程序的开发，所以，对于编译工具的工作原理和工作过程的理解显得尤为重要，通常计算机体系结构的课本都会介绍处理器的指令集和编程模型，并且处理器的芯片厂商也会提供指令集和编译工具的技术手册。

0.4 为什么使用命令行脚本工具进行软件编译

尽管大多数处理器都有配套的带有GUI的开发环境，本设计采用的NIOSII软核也不例外，但是这里仍然采用了命令行脚本式的编译工具链，这是出于以下的考虑：

- 大多数处理器的开发环境均是由上层的图形界面来驱动底层的编译命令进行工作。
- 对于开发环境软件版本升级的情况，脚本式的工程管理具有最好的可移植性。
- 当在一个设计项目的基础上进行修改来得到另一个项目时，使用脚本管理的项目通过修改少量脚本代码即可完成项目的迁移，从而避免了多次点击鼠标的工作。
- bash、make、gcc这样的软件工具在开源世界里有着庞大的用户群体和代码资源。
- 有时，GUI环境会压制一些编译的过程信息，从而使得出现编译错误时，用户面对着一个简单的报错信息无从下手，脚本式的工具链会在终端上打印出编译命令的所有输出字符信息，从而使得用户追查错误的来源变得更加容易。

对于初学者来说，bash与make的命令显得晦涩，所以建议先从GUI的界面开始熟悉一款处理器的开发流程，这种GUI式教程通常很容易得到。当你熟悉了GUI的开发流程之后，再尝试使用脚本来管理工程代码文件和目录。

0.5 请了解一下软件的编译过程

使用命令行脚本来管理工程代码会强迫我们理解编译工具的工作原理和流程，而这种对工具的认识是非常宝贵的经验，因为它会让我们建立起相对良好的代码设计习惯，从而避免引入一些稀奇古怪的编译或是运行错误。请记住，编译器也不是完美和无懈可击的，我们需要知道它的原理，只有给它正确的、清晰的输入，我们才能得到正确的输出。

1 系统功能与顶层结构

1.1 系统功能

如图1所示，本系统由一块型号为DE0或DE2-115的FPGA开发板、一个VGA接口的显示器组成，FPGA开发板内部自行产生数字信号波形，再由其内部的软硬件协同进行处理与分析，最后在VGA显示器上绘制信号分析结果。该系统通过两个拨码开关对电路进行控制。

- SW0，拨到“上方”，电路进入复位状态

- SW0，拨到“下方”，电路进入工作状态
- SW1，拨到“上方”，信号触发捕获被使能
- SW1，拨到“下方”，信号触发捕获被禁止

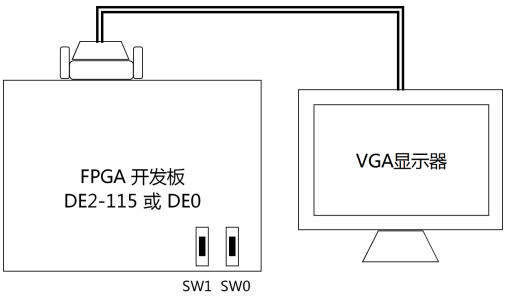

图 1: 数字信号分析系统功能结构

系统完成后信号分析的效果如图2所示

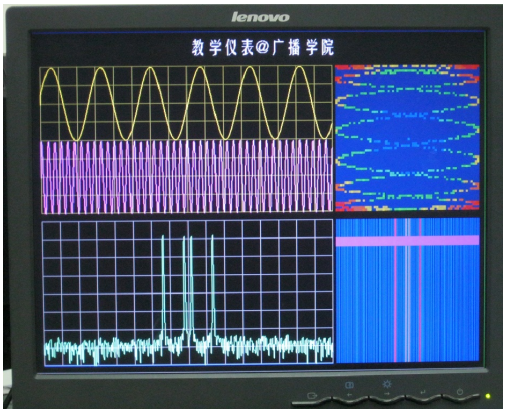

图 2: 系统效果图

1.2 FPGA内部电路结构

如图3所示，FPGA内部电路由以下模块构成

- 信号生成模块：根据处理器的配置，生成2通道频率、相位可独立调节的正弦信号。
- 信号采集模块：外部拨码开关使能开启时，根据处理器的配置，根据触发条件对输入信号进行采集，缓冲区采满后，置位标志寄存器，通知处理器读取数据。
- 处理器软核：配置各个外部模块，读取采集的信号数据，对数据进行分析和显示适配处理，然后将显示数据写入显示器适配模块。
- 显示器接口模块：将处理器写入的波形和显示数据，根据VGA时序绘制在显示器上。
- 时钟方案：外部输入晶振50MHz，作为CPU和显示器接口模块的时钟，信号生成模块工作在80MHz的时钟下，信号采集模块为双时钟，用于跨域时钟域。FPGA外部的SDRAM时钟由PLL移相生成。

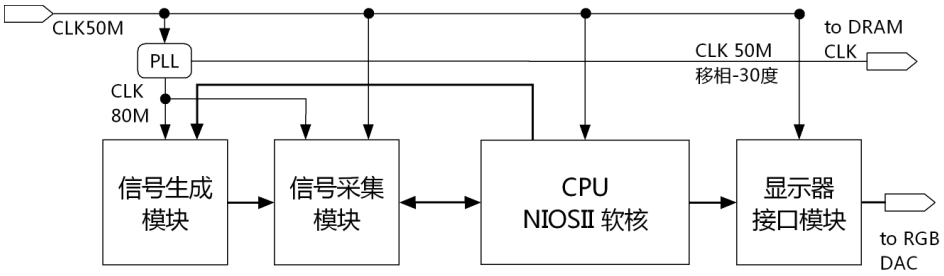

图 3: FPGA内部电路结构

1.3 软件结构

本设计的软件工作流程如图4所示，系统上电复位之后，处理器对各个软件、硬件模块进行初始化，然后进入工作循环。在工作循环中，首先对信号产生电路的参数进行配置，然后启动信号采集模块，并且监测信号采集模块的状态位信息，一旦数据缓冲区采满，则读取所采集的信号数据，进行后续的时域、频域信号分析运算，由于信号的采集长度，分析长度，以及显示器绘图区域的像素尺寸可能存在出入，所以为了适合显示环境，还需再对信号分析的结果数据进行显示适配计算，最后将数据写出至显示器接口电路。

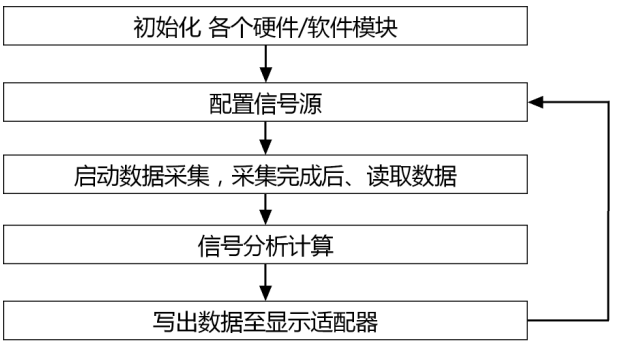

图 4: 软件总体流程

2 硬件设计

2.1 处理器软核配置

Altera提供的处理器集成开发环境用于对处理器软核的周边电路进行配置，本设计需要用到以下的控制器电路。

- SDRAM控制器：用于访问FPGA外部的SDRAM，保存软件的数据和代码。
- PIO接口电路：用于控制FPGA外部的LED灯。
- JTAG UART：用于支持NIOSII 的字符调试终端。
- 总线接口控制器：用于把处理器的总线读写相关信号引出到外部。

2.2 总线接口电路

NIOSII 处理器采用avalon MM 格式的总线，桥接电路将处理器的总线读写信号引出到处理器的外部，外设中的处理器接口电路负责把处理器端向各个地址的读写动作和数据映射到具体的各个外设部件这样做的好处是：自定制的电路部分和FPGA软核供应商提供的电路进行解耦，从而提高整体设计的可移植性。如图5所示

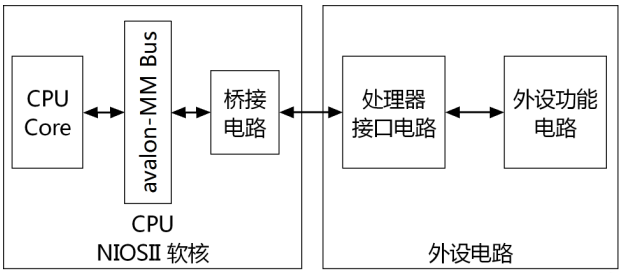

图 5: 处理器总线接口电路结构

2.2.1 avalon总线外部桥接电路

总线外部桥接电路按照avalon-MM总线接口定义的时序，把处理器的读/写动作的使能、地址和数据信号映射到处理器硬件环境之外，为了提高电路的时序性能，需留出足够的流水线节拍裕量，此处设定处理器的读取延迟周期为4，并且在桥接电路内部对输入、输出信号添加流水线进行数据缓冲。处理器的总线读写时序如图6所示

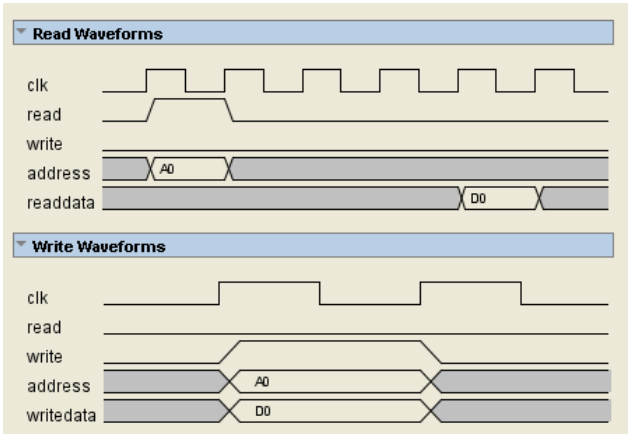

图 6: avalon总线读写时序

2.2.2 处理器读、写寄存器

对于小规模的数据，例如硬件电路的状态或是使能信息，使用存储器映射寄存器的方式和处理器进行通信，该电路检测处理器的读、写地址与使能信号，若与其映射地址相符合，则与处理器进行数据交互。

由于处理器和外设电路可能由不同频率的时钟驱动，此类电路需要解决跨时钟域传输数据的问题，具体方法为在目标时钟域对源时钟域送入的数据进行两级级联的触发器缓冲，当两级触发器的采集数据相等时，说明源时钟域送入的数据被稳定采集，则更新目标时钟域的触发器。

2.2.3 处理器读写RAM

对于处理器和外设之间向量数据帧传输，使用FPGA内部的双口RAM实现，当前FPGA内嵌的双口RAM支持独立的读、写时钟，从而能够有效的实现跨时钟域传输数据，处理器对RAM的访问方式分为两种，一种是随机访问，处理器的地址直接和RAM的地址一一对应，可以实现随机的读、写，该方式的缺点是需要占用较多的处理器寻址空间，通常用在样点数为几千规模的情况。

另一种处理器访问外设存储器的方式为连续地址突发式访问，该种方式只需要一个配置地址、一个数据地址、共两个外设地址。以数据写入为例，处理器向配置地址写入RAM选通，突发写入的初始地址等配置信息，配置完成后，处理器向数据地址连续写入数据，这些数据会被写入到目标RAM的相应地址。电路中使用一个选择器来选通各个寄存器或是RAM的写端口，另外需要使用一个计数值可加载的计数器，该计数器对处理器的读/写使能信号进行计数来生成RAM的地址。电路结构如图7所示

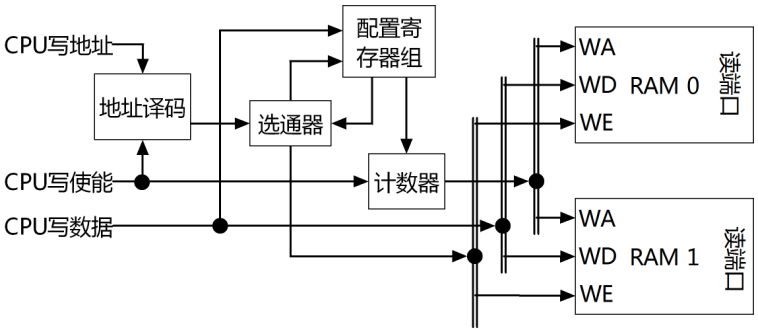

图 7: CPU突发写入RAM接口电路结构

2.3 处理器外设电路模块设计

2.3.1 信号生成

该模块由2个DDS核构成，处理器通过写入寄存器传送配置参数，根据处理器的配置，电路生成两通道正弦信号，每通道正弦信号的频率、相位可以独立设定，两路正弦信号构成一路复数信号输出。

2.3.2 信号触发与采集

该模块在满足触发条件的情况下，对输入的信号进行采集，采集的信号数据被写入缓存RAM，缓存采满后，置位标志位通知处理器读取数据，在采集数据填充缓存RAM的过程中，电路不在响应外部的触发条件，直至缓存采满后，电路重新捕捉满足触发条件的信号。

2.3.3 数据显示绘图模块

该模块用于接收CPU写入的绘图数据，并将其绘制到VGA接口的显示器上，显示参数设定为800x600@60Hz，像素时钟为50MHz。模块内部由1个VGA同步信号发生电路；3个栅格绘图电路；3个信号样值绘图电路及2通道各512点x7比特的时域信号样值RAM、1通道512点x8比特的复信号双边幅度谱分贝样值RAM；1个李萨如图形64x64彩色点阵绘图电路及4096x4比特像素RAM；1个频谱彩色瀑布图256x16彩色点阵绘图电路及4096x4比特像素RAM，其中李萨如绘图电路将图像点阵在水平和垂直方向各放大4倍，频谱瀑布绘图电路在垂直方向放大16倍，两图像点阵的显示尺寸均为256x256 像素，且需要内置的色彩合成电路将像素色彩的存储值映射为显示值。该模块电路结构和显示器的布局如图8所示。

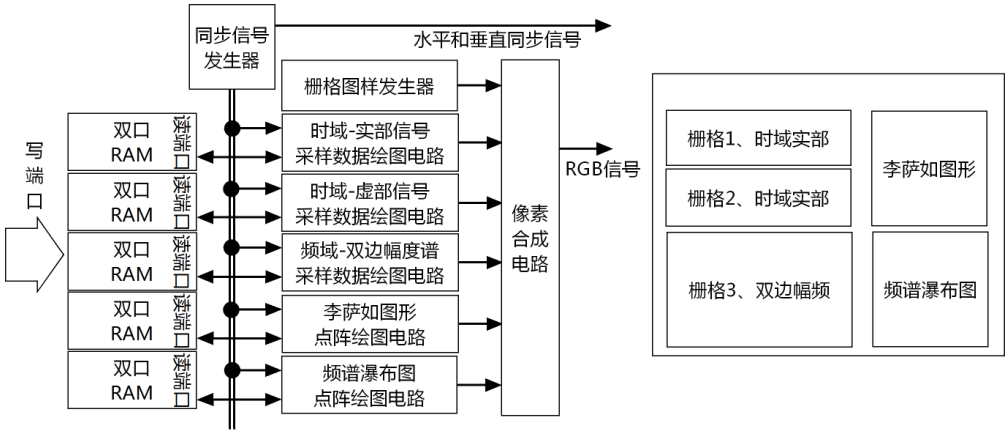

图 8: 数据显示绘图模块电路结构

关于该VGA绘图模块的详细说明，请参考之前发布的参考设计文档，“LOGIC VGA电路模块设计文档”。

3 软件设计

系统上电复位之后，处理器对各个软件、硬件模块进行初始化，然后进入工作循环。在工作循环中，首先对信号产生电路的参数进行配置，然后启动信号采集模块，并且监测信号采集模块的状态位信息，一旦数据缓冲区采满，则读取所采集的信号数据，进行后续的时域、频域信号分析运算，由于信号的采集长度，分析长度，以及显示器绘图区域的像素尺寸可能存在出入，所以为了适合显示环境，还需再对信号分析的结果数据进行显示适配计算，最后将数据写出至显示器接口电路。

3.1 系统初始化

处理器上电工作进入main函数后，首先进行硬件和软件初始化工作，使得各个硬件电路模块进入确定的工作状态，以及初始化软件函数需要使用的常量系数表。

3.2 信号源配置

在每次工作循环的遍历过程中，需对信号源进行配置，该配置是一个两层迭代的过程，其中通道1正弦信号的频率和相位固定，然后逐次改变通道2的相位和频率，内层迭代按照某个角度步进改变信号的相位，待相位从0到360度扫描一周后，在外层迭代过程中再对正弦信号的频率进行步进累加，待信号频率达到最高值后，再回绕至初始频率继续迭代。

3.3 数据触发与采集

该过程首先监测硬件采集电路的工作状态，如果信号采集电路的空闲状态下开启信号采集的触发使能，信号采集电路的使能信号工作于上升沿触发模式，对其配置寄存器的一次置位/清零操作将导致信号采集电路的一次数据触发与帧采集操作。一旦数据缓冲采满，硬件将帧号计数值加1，软件从相应寄存器读到帧号增加获知新的数据帧采集完毕并将其读取到软件的数据缓冲区当中。

另外，软件带有超时记录功能，如果信号不满足触发条件，硬件久未捕获到新的数据帧，软件则用前一帧数据进行绘图，从而帮助仪器显示的连贯性。

3.4 信号分析

信号分析模块工作为对输入的复数数据帧进行快速傅里叶变换计算，并且将计算的结果转换到分贝尺度，然后将数据移动至零频居中双边谱格式，最后根据信号显示的栅格尺寸将数据缩放至20dB/div的数值尺度。

3.5 信号显示

本设计中有5处信号显示区域，2处时域波形矢量，1处双边幅度谱矢量，1处李萨如图形点阵，1处频谱瀑布图点阵。

对于时域波形，可直接从信号的触发点顺序将数据写入显存。对于频谱矢量，需要进行显示适配，例如将2048点的频谱计算结果写入到水平尺度为512点的显存中，则需要对原始数据进行缩放，为了便于用户观察信号频谱的变化，采用“峰值保持”的抽取策略，即，设抽取因子为N，则在原始的2N个数据中，搜索出最大值与最小值2个点作为输送至显示缓存相应位置的数据。

对于李萨如图形的点阵绘图，首先将I/Q数据值缩放至图像点阵的水平和垂直尺度范围，另外需计算出复数样点的模值并根据该模值映射出绘制样点的伪彩色以显示信号强度。

对于频谱瀑布图，需首先将计算得到的频域样点数据按照瀑布图的水平尺度进行缩放，然后进行信号强度的伪彩色适配，计算出当前瀑布图行的彩色样点数据并更新相应行像素。

4 开发环境与工具链

4.1 目录结构

- /doc：文档
- /hardware_cpu：处理器相关的硬件电路，以及顶层硬件设计
- /hardware_logic：自行设计的处理器的外围电路
- /software：软件，包含2个子目录
  - /app：应用程序源代码
  - /bsp：板级支持包代码

4.2 硬件逻辑设计

- HDL编译工具，quatus 10.1
- SOPC工具，SOPC builder

4.3 软件编译与下载调试

使用altera提供的编译命令脚本create-this-bsp和create-this-app 进行编译，使用nios2-configure-sof和nios2-download -g下载软硬件，使用nios2-terminal 进行调试。

- 生成bsp配置文件：运行software/bsp/create-this-bsp 生成settings.bsp文件，运行GUI工具中的bsp editor 修改settings.bsp的设定，根据需求生成新的配置文件。
- bsp库编译：再次运行software/bsp/create-this-bsp 编译生成bsp的库。
- app编译：运行software/app/create-this-app 编译应用程序
- 下载软硬件：使用nios2-configure-sof xxx.sof下载硬件电路，使用nios2-download -g xxx.elf 下载目标程序并立即运行。
- 软件调试：新开启一个终端，使用nios2-terminal 启动电路板的调试终端。

5 推荐的参考书目

编写这个参考设计的动机很简单，就是希望展示一下基于处理器的软硬件联合设计。我相信，在我们能够用Verilog在FPGA上设计出一些电路，以及对照着厂商的指导手册，使用它们提供的处理器软核做出一些设计之后，很多同学会和我有同样的困惑，如何才能够进一步的深入？

个人观点，无论是verilog，FPGA，处理器软核或者一颗真正的处理器芯片，只不过是个工具，学会使用它们只是个开始，接下来的事情分为两个方面，一是要明确以后用这些工具做些什么事情，这一点我帮不了你，因为我不知道你想做什么。我是要用FPGA、处理器、数据转换器和模拟器件做电子设备，带着小

朋友们用这些自制的设备做无线电信号处理的实验然后一起去吃火锅的。另一方面，要明确这些工具的原理和其后面的学科理论，这一条我能够帮你的是推荐几本参考书。

先哲说过，“最复杂的数字电路莫过于处理器”，处理器是我们人类信息化浪潮的核心组件之一，人类为了设计出更好的处理器，专门在开设了一个学科，叫做“计算机体系结构”，所以读一读这方面的教科书对我们设计数字电路和使用处理器非常有帮助，并且由于我们使用处理器的时候要对它进行编程，所以了解一下高级语言编译器的工作原理也很有必要。

首先推荐几本科普性质的读物，左飞，《代码揭秘：从C/C++的角度探秘计算机系统》，这个书比较入门，告诉你处理器、操作系统、编译器如何协同工作的。并且用Visual Studio 做的例子，很容易照着做练习。然后，作为故事书，看看万木杨，《大话处理器:处理器基础知识读本》这个书用非常平易近人、通俗易懂的文笔介绍了处理器体系结构的原理和实例。以上这两位，虽然不是什么学术权威，但人家是真正在科技公司的一线研发队伍里工作了好多年的工程师，是真正的实践过的经验人士，所以，他们从实践中总结和提炼出的文字也具有很高的可读性和启发性。

然后，该看正规教科书了，计算机体系结构方面，我推荐《计算机组成与设计硬件/软件接口》，作者是David A.Patterson, John L.Hennessy，斯坦福大学的教授和校长，RISC结构的创始人，属于目前还活着的泰山北斗级别的人物，这俩人还写了本书，《计算机体系结构：量化研究方法》此书比较艰深，如果你要改行去设计CPU或是要编写系统级的非常高效率的软件的时候可以阅读，一开始先别急着读，会熬不住的。

另外，还有一本名著，Randal E.Bryant / David R.O’Hallaron 合著的《深入理解计算机系统》，此书被誉为“价值超过等重量黄金的无价资源宝库”，我个人感觉是，此书是从程序员的角度来审视讲解计算机的软硬件系统，而斯坦福的两位爷是从硬件工程师的角度来描述处理器的。

6 后记

我在本科母校的一个研究所里面度过了自己的研究生岁月，回想起来，我的研究生时光可以用“快乐且迷茫”来形容。

快乐是因为我的导师和其他老师们给我们创造了非常良好的科研环境，充足的科研经费，先进的实验设备和课题任务，以及相对自由且宽松的工作学习气氛。

在这样好的学习环境里我却仍然感到迷茫，究其原因是因为电子工业的飞速进展，相对于老师们读书的时代，我们可以选择更多的器件、更多的软件工具，并且面对着更加复杂的技术应用，这一切让人眼花缭乱、无从选择。

当时我经常被一个问题困扰，为什么我在操纵这些新式的软件工具和硬件电路方面比我导师熟练，然而在对于电子系统和技术路线的理解这一方面，我在他的面前却是如此幼稚。后来，直到有一天，我在和他闲聊的时候得知，他的老师年事已高，曾经一度连计算机都不会用，但是却能够指导研究生用工作站和EDA软件来设计集成电路，究其原因，得益于我们这位祖爷爷年轻时曾经在扩散区、沟道、晶体管这种层面设计过相当数量和规模的电路，于是我明白了，基础知识和基本功是多么的重要。我们不能满足于使用最新式的器件和工具把设计做完就好，要明白这些工具和器件的原理，必要的时候，我们甚至要自行设

计这样的工具和底层模块。

所以，在毕业后留在这个实验室工作的今天，我已经不再有那样的迷茫，因为我知道了，无论技术如何发展，那些最底层的电路和算法的价值依然存在，因为它们经过几代科学家和工程师的积累，凝聚了无数的智慧与汗水，这是我们电子工业最宝贵的财富，只要掌握了它们，我们就可以从容的面对一轮又一轮的技术浪潮与变革。

于是，不知怎的，我便得了这般小小勇气，激励我在科研工作与家务劳动的余暇时光里，制作如此的参考设计与文档，若您和曾经的我有着同样的迷茫，愿这份参考设计对您能够产生些许的帮助，祝好。