原创性声明

郑重声明：此篇题为《基于Avalon总线的8051MCU IP核的设计》的论文是作者在导师的指导下，于武汉大学攻读硕士学位期间，进行研究工作所取得的成果。根据作者所知，论文中除了参考文献列举的地方外，不包含其他人已经发表或撰写过的研究成果。

作者签名：王安文 倪奎 旷捷 程方敏

导师签名：黄启俊 常胜

撰写日期：二零零九年六月十八日

基于 Avalon 总线的 8051MCU IP 核的设计

作者：王安文 倪奎 旷捷 程方敏

导师：黄启俊 常胜

（武汉大学物理科学与技术学院电子科技系，武汉，430072）

摘要：本文设计了一款基于 Avalon 总线的 8051MCU IP 核。它支持 MCS-51 指令集，优化内部的结构，通过采用流水线技术、指令映射技术、指令预取技术、微代码技术等极大的提高了 IP 核的工作速度，使 IP 核在 100MHz 时钟下，能够单周期执行一条指令。本设计使用 Modelsim 软件完成了功能仿真和时序仿真，并在以 Altera 公司的 Cyclone II FPGA 芯片为核心的 DE2 开发板上完成了硬件验证。

关键词：MCS-51、Avalon 总线、流水线

Design of 8051 MCU IP Core Based on Avalon Bus

Authors: Wang Anwen, Ni Kui, Kuang Jie, Chen Fangmin

Tutors: Huang Qijun, Chang Sheng

（Department of Electronic Science and Technology,College of Physics Science and Technology, Wuhan University,Wuhan,430072）

Abstract: This paper introduces a design of Avalon-bus-based 8051 MCU IP core, which is compatable with the MCS-51 instruction set. To improve its performance, the IP core is optimized by many technologies, such as pipelining, instruction mapping, instruction pre-fetch, micro-programmable technology and so on. As a result, the speed of this IP core is greatly increased, so that it can handle one instruction in a signal cycle. The function simulation and timing simulation of this design are complished by Modelsim, while the whole design is verified on the Altera DE2 development board.

Keyword：MCS-51, Avalon Bus, Pipeline

1 引 言

随着电子设计技术的飞速发展，基于 IP 核重用的 SoC 设计技术逐渐成为主流设计技术。市场上出现了各种各样的 IP 核，例如处理器核、DSP 核等，这些 IP 核的应用极大地提高了电子设计的效率。8051 作为一个应用广泛的 8 位单片机，出现了一些对应的 IP 核。例如 Digital Core Design 提供的 DR8051 采用 RISC 架构，工作速度比原来的 8051 快了 6.7 倍；文献[1]介绍的 Cygnal 公司的 CIP-51 采用流水线结构，工作最高时钟 25MHz，70%指令能够单周期执行；文献[2]介绍的 Synopsis 公司的 DW8051-core 在 4 个周期执行一条指令。这些 IP 核增加了 MCS-51 系列单片机的应用灵活性，不过这些 IP 核的还是沿用传统的 51 接口，即并口 P0、P1 等，或者使用 16bit 地址总线和 8bit 的数据总

线。总线协议非常简单，在复杂的应用环境中不能满足要求，因此有一些设计者在 IP 核中添加了一些 SoC 标准片上总线系统，例如文献[3]介绍了在 DW8051 核中添加了一个 AMBA 总线接口，方便了系统集成。这样可以通过标准总线方便地进行系统集成，避免了接口之间胶合逻辑的设计，有效地提高了设计的效率。虽然目前已经有比较成型的 51 IP 核，但是在速度和 IP 核接口的通用性方面还有相当大的工作要做。基于上面的考虑，结合 Altera 公司的 Avalon 总线协议，设计了一款带有 Avalon 主端口的 8051 MCU IP 核。它主要有两个特点：

1. 8051 MCU IP 核的接口设计成 Avalon 总线[4]接口，这样使 IP 核可以通过 Avalon 总线方便快捷的集成到系统中去，而不需要再添加任何的接口逻辑。另外采用 Avalon 总线接口可以直接使用 Altera 公司提供的 SOPC Builder 工具来集成系统，从而进一步降低了系统集成的难度。
2. 8051 MCU IP 核的峰值工作速度是单周期执行一条指令。为了达到这一目标，在 IP 核的设计中，主要采用流水线结构来执行 51 指令，提高指令的执行速度；同时采用指令映射技术和指令预取技术，提高指令的读取速度。通过优化技术或结构，在保持高频率的时钟条件下，提高了 8051 MCU IP 核的工作速度。

2 设计思路

51 指令[5]是 CISC 指令系统，它具有指令多、操作复杂、指令长度不同等一系列特点，因此优化起来比较复杂。为了能够设计一款高速带有 Avalon 总线的 IP 核，对 IP 核的实现架构做了如下 5 个优化技术：

1. 流水线技术

在 IP 核中将每一条指令都划分成 6 级流水线：指令预取、译码、读操作、获取数据、执行 ALU、回写结果，见图 1。

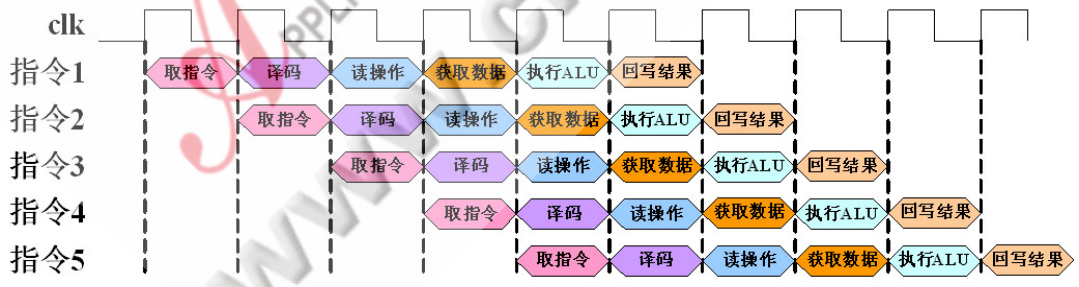

图 1 **IP** 核的流水线划分

- 指令预取：预先将外部程序存储器的指令取到 IP 核的指令队列中，这样可以提前准备好指令。
- 指令译码：翻译指令，根据指令的机器码，翻译成各个模块的控制信号。这些控制信号完成指令的执行。
- 读操作：本指令中需要的操作数在内部 RAM 或者外部存储器中，这样需要发起一个读操作来取出数据。

- 获取数据：将指令中处理的操作数输出到数据总线中去，这些操作数的来源有多种渠道：内部 RAM 的数据输出口、通用寄存器、某些特殊寄存器、外部存储器的数据。
- 执行 ALU：发出执行 ALU 指令，对数据总线上的数据做运算处理。
- 回写结果：将 ALU 运算的结果回写到存储器中（主要针对内部 RAM 的操作，外部 RAM 的读写需要一些额外的延时）。此时发出写操作。

根据上面结构的特点，将所有的指令细化，分别用独立的部件完成每一级流水线的功能。另外这样的流水线结构需要考虑数据的相关性问题，即相邻指令之间涉及到相同地址数据的读写时，数据来不及更新。这个问题会造成数据处理错误。解决办法是内部前推技术，即内部设计有一个专门通道，在出现数据相关时，将回写的数据通过这个专用通道直接将其输出下一个指令的输出数据口。

2. 存储器分页机制

因为 IP 核内部只有 16bit 的地址线，访问 32bit 的地址空间。为了充分的利用存储空间，采用分页机制，即在 IP 核内部添加了段地址寄存器，用来标注页面。这里设计了两个段地址寄存器：程序段地址寄存器 CS；数据段地址寄存器 DS。均是 16bit 的长度。外部存储器的实际物理地址是由段地址寄存器和 IP 核内部的地址线来组合。这种分页机制具有以下特点：

图 2 外部存储器的分段机制

- 每一个页面内的存储空间是 64KB，因此能够完整的被 16bit 地址线寻址，所有页面空间都是独立的。页面之间不会出现重叠。
- 在组成物理地址时，只需将段地址作为高 16bit，IP 核内部地址线作为低 16bit 即可，地址翻译简单。
- 用户自己保证存储程序的页面和数据的页面不能相同，这样避免产生冲突。

3. 指令映射技术

在设计中，一般将程序存储空间开辟在 Flash 等非易失性存储器内，这样程序就可以保存起来。但是这种非易失性存储器的读周期太长（一般会在 70ns 以上）。因此如果直接从 Flash 存储器中读指令会花费很长的时间。解决办法是指令映射技术，即在程序开始执行之前，先将 Flash 的程序搬运到高速的易失性存储器或片上存储器中，然后再直接从高速存储器中读取指令执行。8051 MCU IP 核

的程序存储器空间只有 64KB，很容易在高速存储器中分配这样的一块空间。见图 3，系统将片上存储器的 2KB 的空间作为了程序块映射的目标地址。

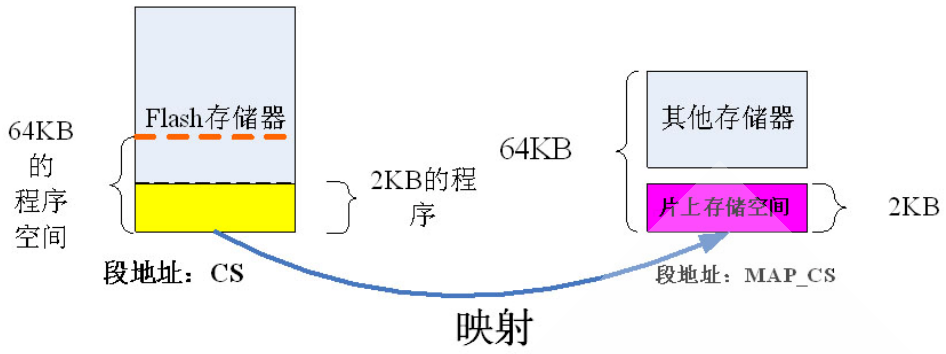

图 3 程序空间映射

这种方式需要定义两个 16bit 的段寄存器：

- CS：代码段地址寄存器；
- MAP_CS：映射后的代码段地址寄存器，表示会将代码映射到此段地址空间内；

4. 指令预取技术

为了能够使流水线流畅的执行，需要在每一个周期都送出一条新的指令。为此需要将待执行的指令提前准备好，这里设计了指令预取技术，即将要执行的多条指令提前读入，存入指令队列中。

5. 微代码技术

采用 ROM 结构直接将控制信息存储起来，然后使用指令的机器码寻址来获取控制信息。这样设计者可以通过更改 ROM 内的信息达到修改控制的目的，另外这种 ROM 寻址的方式可以有效地降低组合逻辑大小，极大地提高工作速度。针对 CISC 指令系统，由于指令较多，如果采用硬件直接译码会出现很大的译码逻辑，严重影响速度，而且非常占资源。在 FPGA 中采用微代码技术可以使用器件内部的 RAM 资源，同时又可以有效节省宝贵的 LE 资源。

上面的五个技术都是从不同的侧重点来优化正 IP 核的指令执行速度：流水线技术是在执行方面优化速度，达到单周期执行完指令的目的；存储器分页机制、指令映射技术和指令预取技术都是在取指令方面优化速度，能够及时地将执行的指令准备好；微代码技术是在译码方面优化速度，能够快速地对指令译码，获取指令的控制信息。

3 系统实现

3.1 系统框图

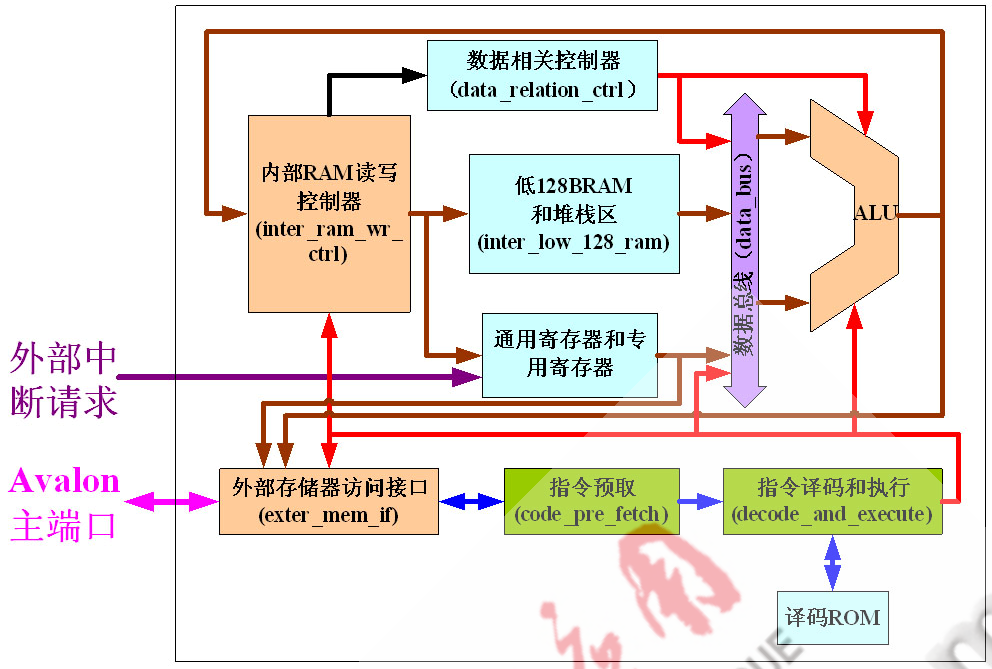

图 4 8051 MCU IP 核的系统结构框图

8051 MCU IP 核的系统结构见图 4，由 10 个功能模块组成，外部接口有中断请求、Avalon 主端口。这 10 个模块的功能如下：

- 内部 RAM 读写控制器：控制对内部 RAM、堆栈区、通用寄存器区进行读写，它的控制信号来源于指令译码和执行模块，同时它还会将当前执行指令的源地址和目标地址发送给数据相关控制寄存器。
- 数据相关控制器：数据相关控制器接收当前指令的源地址和目的地址模块，然后判断是否出现数据相关，给出判断结果。
- 低 128B RAM 和堆栈区：定义了 IP 核内部 RAM 的低 128B 空间，另外定义了一个独立的堆栈区。
- 通用寄存器和专用寄存器：定义了通用寄存器 Rn 和一些专用寄存器。
- 数据总线：定义了数据总线，在获取数据时，会将数据输出给数据总线。
- ALU 单元：运算单元，处理各种运算。同时里面还包含有一些独立的寄存器 A、B、PSW 等。
- 外部存储器访问接口：带有 Avalon 主端口，用于访问 Avalon 总线。可以用于读取指令、读/写数据。
- 指令预取：预取指令，同时计算新的 PC 值，并且将待处理的指令输出给后面的译码模块。
- 指令译码和执行：将待处理的指令的机器码作为地址查找译码 ROM，获取该指令的控制信息，然后将此指令发送给执行模块。执行模块会将控制信息输出，用来直接控制 IP 核各个流水线的工作过程。
- 译码 ROM：内部保存有各个指令的控制信息，指令的译码过程就是查表过程。

在接下来的内容中，将介绍主要模块的实现及一些模块的验证结果。

3.2 主要模块的实现

8051 MCU IP 核的设计核心思想是实现六级流水线结构，因此 IP 核的内部通过几个主要的控制模块来分别实现每一级流水线的工作，见图 5。下面将分别描述这几个模块的具体实现。

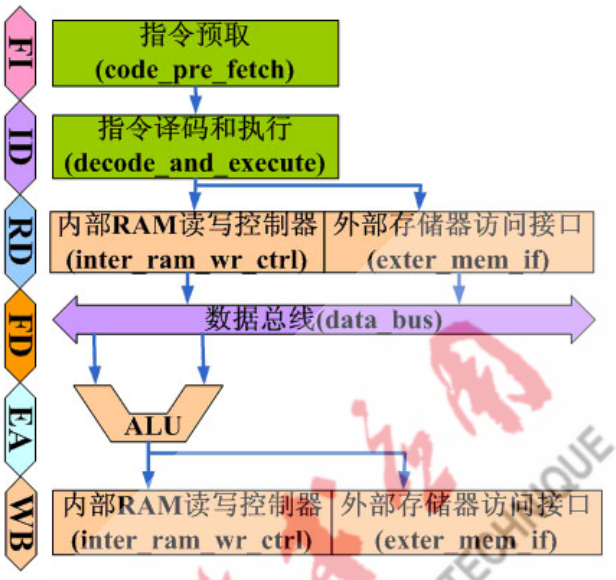

图 5 流水线在 8051 MCU IP 核内部的实现

3.2.1 8051 MCU IP核的指令预取

指令预取是具有缓冲功能的部件，将以后需要执行的指令预取出来，保存在队列中。具有以下一些技术特点：

- 内部包含容量是 10×42bit 的队列。最多能够保存 10 条指令信息，每一条信息包含有 24bit 的指令数据、16bit 的下一个指令的 PC 值、当前指令长度（以字节表示）。
- 模块内部将输入的指令做初步译码，得知指令长度。然后将整条指令那个拼凑在一起，然后存入指令队列中。
- 外部存储器访问模块一次会为此模块提供 4B 的程序代码。提供的存储代码会立即进行初步译码，得出指令的长度和类型，会出现以下几种情况：
  - 如果是条件跳转类指令：将指令直接存入队列，同时将此指令送给分支预测工具处理，获取预测的目标地址。
  - 如果是直接跳转类指令（ACALL、AJMP、LCALL、LJMP、SJMP）：将指令直接存入队列，同时计算出跳转的目标地址。
  - 如果是间接跳转类指令（JMP、RET、RETI）：将指令直接存入队列，同时停止指令预取工作，直到该指令有了执行结果时，才知道目标地址，此时再读取目标地址的程序。
  - 如果是其他类型的指令：将指令直接存入队列。
- 如果是中断请求信号，找到中断跳转指令。同时将指令队列清空，此时根据中断跳转指令的目标地址开始读取正确的指令。

总结上面的功能可以将指令队列划分成 4 个子模块：指令读取（code_fetch）、地址计算(pc_count)、

指令队列(code_queue)、指令输出(code_output)。见图 6 所示。

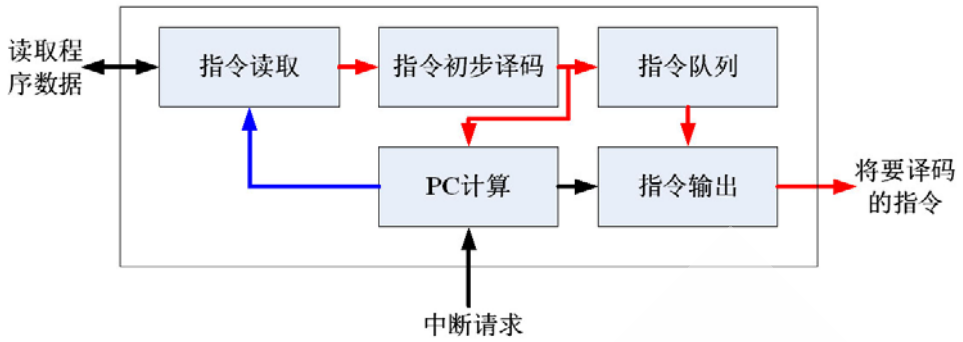

图 6 指令预取模块的内部结构

- 指令预取模块：读取外部存储器，获取指令，对指令进行初步译码，然后将其输出；
- 地址计算模块：对 PC 值进行控制，考虑各种跳转状态、中断状态的情况；
- 指令队列模块：将前面的指令预取的输出指令存入指令队列中，同时将队列前面的指令输出；
- 指令输出模块：读取指令队列的输出模块，另外如果出现中断请求时，需要插入中断指令；

指令预取模块实现后的验证结果见图 7，读程序地址 PC 是 0000H，读入程序数据是：12153524H；经过内部处理最后的数据指令是：第一条指令：24H、第二条指令：1535H；还剩下的 12H 不能构成一个完整的指令，因此需要等待下一次读入程序数据。可见达到了设计要求。

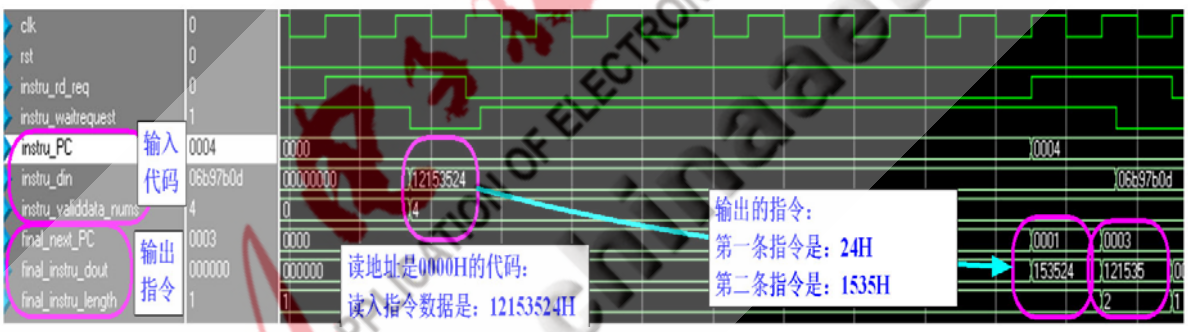

图 7 指令预取模块的仿真结果

3.2.2 8051 MCU IP核的指令译码和执行

本模块的主要功能是接收指令队列模块输送过来的指令，然后以指令的机器码作为地址查找译码 ROM 内的控制信息；然后将这个控制信息送给执行模块输出。它需要考虑一些特殊情况，主要分为两部分：碰到特殊指令时；特殊指令完成后出现特殊情况了

- 碰到特殊指令：需要考虑以下 4 类指令：
  - 考虑 Ri_dirty 信号的指令： ADD A, @Ri ADDC A, @Ri ANL A, @Ri CJNE @Ri, #data, rel DEC @Ri INC @Ri MOV A, @Ri MOV direct, @Ri MOV @Ri,A MOV @Ri direct MOV @Ri #data ORL A, @Ri SUBB A,@Ri XCH A, @Ri XCHD A,@Ri XRL A, @Ri
  - 必须在指令执行前有三个 NOP 指令执行，然后在读期间需要插入等待的指令：MOVC A, @A+DPTR MOVC A,@A+PC MOVX A,@DPTR MOVX A, @Ri

- 必须在写期间插入等待的指令：MOVX @Ri, A　　MOVX @DPTR, A
- 其他特殊指令： DIV AB　（前面必须有 5 个 NOP 指令，然后使用 GET_DIV_RLT 指令）

- 特殊指令执行完成后出现了特殊情况：
  - branch_lose 信号有效，表示出现了分支跳转失效。将指令执行模块的内容立即清除，同时停止译码。

本模块的内部结构见图 8，内部由控制、译码、执行模块组成。

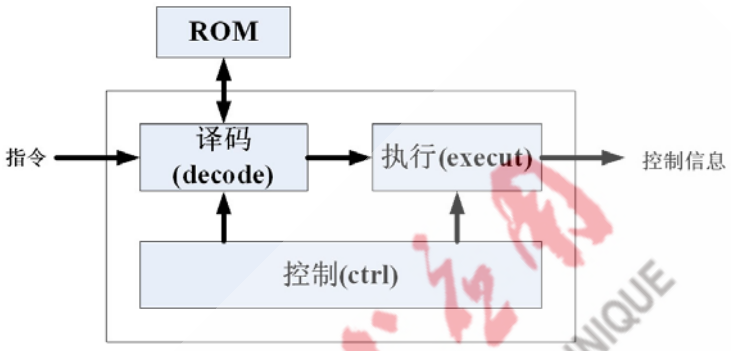

图 8 指令译码和执行模块的内部结构

- 控制模块：对当前的一些特殊指令如涉及寄存器 Ri 寻址的指令、读写外部存储器的指令、除法指令等，进行特殊的处理；另外如果出现了一些异常情况，如条件跳转类指令的分支预测出错时，生成控制信号，停止指令的译码和执行。
- 译码模块：包含有一个读译码 ROM 的控制器，根据输入的指令的机器码直接获取对应的控制信息。
- 执行模块：对译码生成的控制信息进行组合，形成新的控制信息结构，然后直接控制 IP 核中其他控制模块如内部 RAM 读写控制器、数据总线、ALU 等的执行。

指令译码和执行模块实现后的验证结果见图 9。在图中显示指令 XRL direct, #data ，其机器码是 63H，即第一条指令的译码、执行过程。整个指令的译码和执行过程如下：模块接到第一条指令，其机器码是 63H；然后使用这个机器码查表译码 ROM，获取控制信息 220801a043H；然后将控制信息发给执行模块，执行模块会取出控制信息中各个流水线控制信息，然后移位执行，见图中所示的虚线，同一时间输出流水线的控制信号是：第二条指令的读操作、第一条指令的获取数据操作。通过分析整个译码和执行流程能够正确的完成，而且本模块能够一个周期完成一条指令的处理。

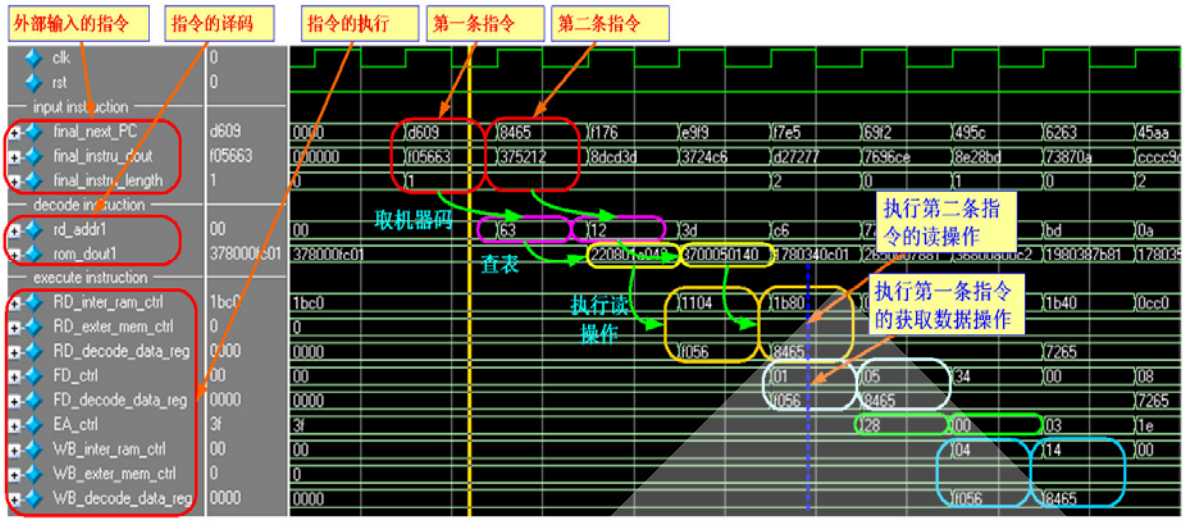

图 9 普通指令的仿真结果

3.2.3 8051 MCU IP核内部RAM读写控制器

内部 RAM 读写控制器依赖外部的控制信号，这些控制信号来源于指令译码和执行模块，其实就是当前指令对内部 RAM 的读写操作。

此控制模块在这些控制信号的作用下，会完成两个工作：读写内部 RAM 区、为数据相关控制器提供本指令的源操作数地址和目的操作数地址。

- 读写内部 RAM 区：内部 RAM 区包括用户 RAM 区、堆栈区、通用寄存器区、专用寄存器区。
- 向数据相关控制器提供数据：当某一指令执行读操作的流水线操作时，同时会给出这个指令将要处理的源和目的操作数，用来判断是否出现数据相关。

3.2.4 8051 MCU IP核的外部存储器访问接口

外部存储器访问接口主要包含两种工作模式：

- 初始化模式：处理器刚一上电，会自动完成一些初始化工作，如指令缓存，将 CS 段内的指令转移到 MAP_CS 段内。转移时采用普通读写方式，数据宽度是 32bit，一次读 32bit 数据，然后将数据写入目标存储区。
- 正常运行模式：处理器在正常运行。
  - 在读数据的流水操作中需要读取外部数据时，此接口模块会通过 Avalon 总线向外部发起一次读操作，读取一个字节的数据。此操作称读外部数据操作；
  - 在回写数据的流水操作中需要将数据写到外部存储器时，此接口会根据要求发起写操作，将一个字节的数据写到指定的单元中。此操作称写外部数据操作；
  - 如果指令队列出现了足够的空间时，需要预取新的指令来填充，可以通过这个接口模块来读外部指令存储器中的指令，一次读操作可以读取 32bit 的指令值（考虑到地址对齐，可能有些指令已经被处理过了，后面设计中详细描述）。此操作称为预取指令操作。
  - 接口模块内部还包含对上面三种操作的仲裁器，优先级顺序由高到低依次是：写外部数据操作、读外部数据操作、预取指令操作。

下面重点介绍正常工作模式。模块内部使用一个工作状态机实现正常工作模式，状态机见图 10，复位后进入 NO_OPER 状态下，然后等待初始化状态机进入 READY 状态后才能开始正常工作。在 NO_OPER 状态时，会出现 4 种情况：

- 当同时有读写数据申请时，即 WR_flag 和 RD_flag 同时有效
  - 状态机跳转到 WR_FIR_DATA，发起写传输，然后进入写等待 WR_FIR_WAIT；写完成后，进入 RD_SEC_DATA，发起读传输，然后进入读等待 RD_SEC_WAIT，当读完成后返回 NO_OPER 状态。
- 当只有写数据申请、无读数据申请时，即 WR_flag 有效、RD_flag 无效
  - 状态机跳转到 WR_DATA，只发起写传输，写完成后跳回 NO_OPER 状态
- 当只有读数据申请、无写数据申请时，即 WR_flag 无效、RD_flag 有效
  - 状态机跳转到 RD_DATA，只发起读传输，读完成后跳回 NO_OPER 状态
- 当只有读指令申请时，即只有 instru_rd_req 有效
  - 状态机跳转到 RD_INSTRU，发起读指令传输，然后进入等待 RD_I_WAIT。读完成后跳回 NO_OPER 状态

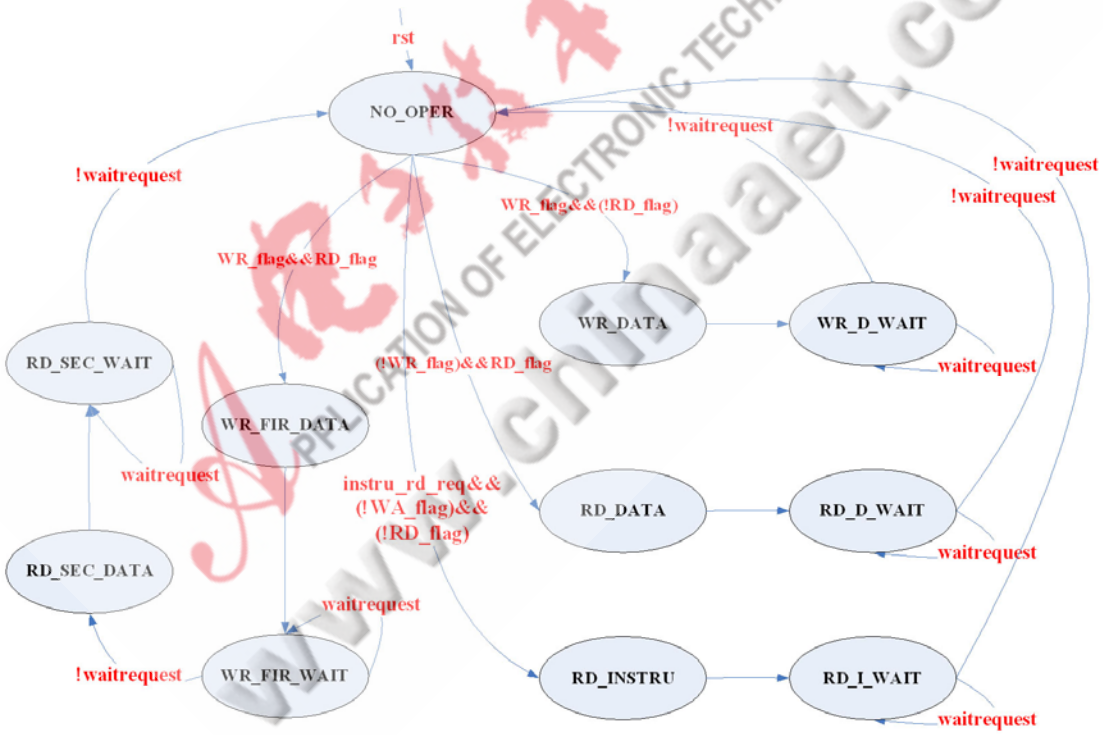

图 10 工作状态机

外部存储器访问接口实现后的验证结果见图 11。图中 RD_exter_mem_ctrl 和 WB_exter_mem_ctrl 均为 1，表示需要同时写外部存储器和读外部存储器。根据状态机的设计，会先进行写操作，然后进行操作，最后返回。观察结果可以看出，仿真结果完全正确。

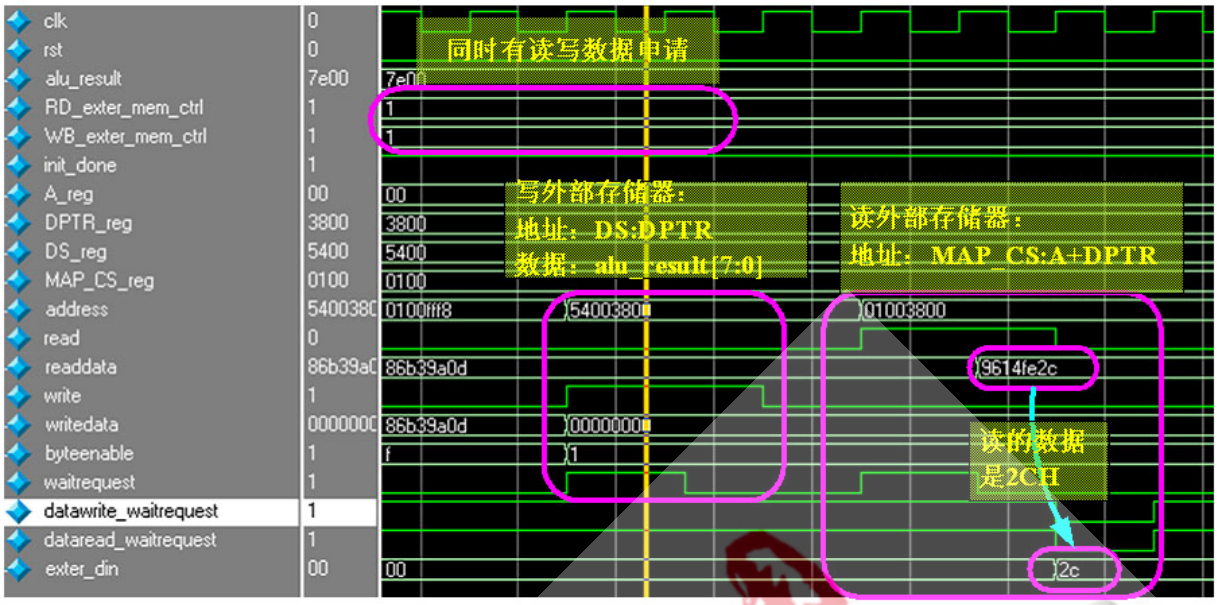

图 11 同时读写外部存储器的仿真结果

3.2.5 8051 MCU IP核的数据总线

数据总线模块是负责选择数据值，将其输出到数据总线上，供 ALU 模块处理。其详细功能如下：

- 在不出现数据相关时，模块根据控制信号 FD_ctrl 来决定哪一个数据源输出到数据总线上；
- 在出现数据相关时，有两种情况：
  - 如果与上一条指令相关，模块会直接将 ALU 的运算结果作为数据源，直接输出给数据总线；
  - 如果与上上一条指令相关，模块将会先将 ALU 的运算结果寄存起来，然后将此寄存的结果输出给数据总线。

3.2.6 8051 MCU IP核的ALU

执行 ALU 的运算功能，包含有各种运算操作（加、减、乘、除、移位、比较等）、数据移动操作。同时还有一些其它功能，如对 B 寄存器的字节寻址访问。

在这个模块中，信号 EA_ctrl 控制着整个 ALU 单元的执行，它是一个 6 bit 位宽的信号，每一种运算操作都对应了一个编码。例如：EA_ctrl 的值是 000000 时，表示将数据总线的数据直接输出；EA_ctrl 是 000001 时，表示将 A 寄存器的值直接输出…… 还有一些其他的操作，这里不再累述了。

本 ALU 模块执行了 8051 MCU IP 核的所有的指令，为了保证速度的优势，需要特别处理其中的乘除法，因为必须避免因为乘除法而出现大的组合逻辑块。因此在设计中，乘法模块采用 FPGA 内部的 DSP 资源；而除法模块采用 Altera 提供的 IP 核，采用三级流水线的方式处理除法模块。这两种方式基本可以达到在 150MHz 条件下完成运算的目的。但是对于除法需要多耗费 3 个时钟周期来完成流水线处理，因此在执行除法指令时需要做如下特殊处理，见图 12。

- 提前插入两个周期的 NOP 指令，目的是确保寄存器 A 和寄存器 B 的值都是最新的；
- 然后执行三个 NOP 指令，目的是让除法器完成流水线除法；
- 执行附加指令 GET_DIV_RLT，目的是将流水线除法的输出结果直接锁存到寄存器 A 和寄

存器 B 中。

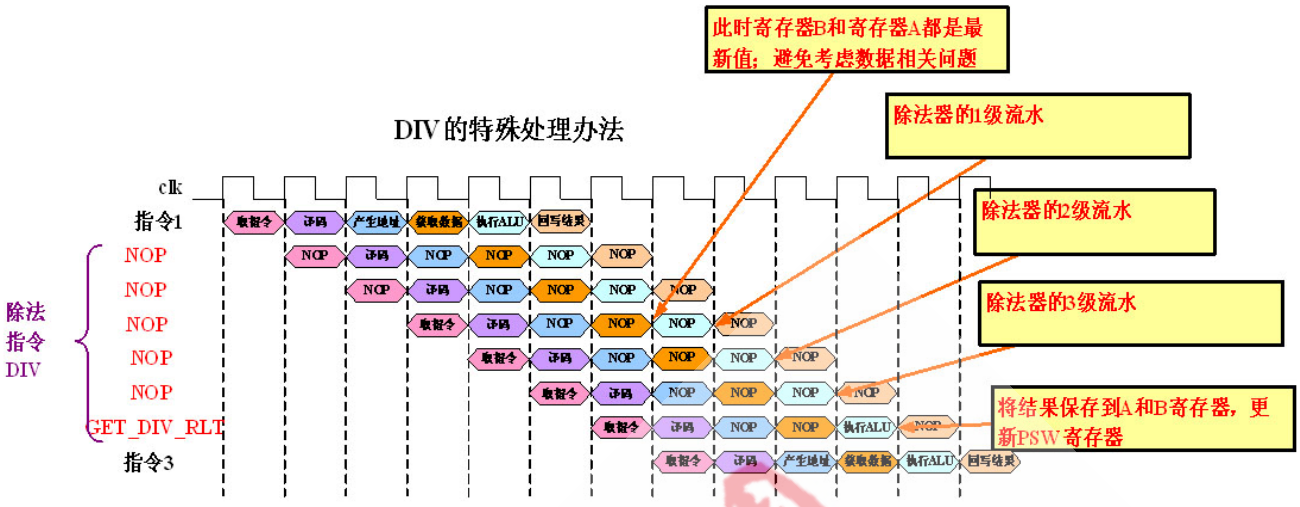

图 12 除法的特殊处理

3.3 系统综合结果

8051 MCU IP 核在 FPGA 上的综合，采用的综合工具是 Quartus II Version 9.0(32bit)；约束的目标器件选了两个 Cyclone 和 CycloneII。具体综合结果见表 1。

表 1 8051 MCU IP 核的综合结果

器件	资源消耗	最高工作时钟
Cyclone：EP1C6T144C6	逻辑单元：2919(49%) 存储器：12266bit(13%)	81.38MHz
CycloneII：EP2C35F672C6	逻辑单元：3343(10%) 存储器：12266bit(13%)	106.78MHz

这里还提供了一些其他 51 IP 核的综合结果，见表 2

表 2 其他 8051 IP 核的综合结果

IP 名	器件	资源消耗	最高工作时钟
DR8051	EPF10K100E	逻辑单元：2049	58MHz
DW8051	CycloneII：EP2C35F672C6	逻辑单元：3087 寄存器：736	80.31MHz

与本设计的 IP 核比较，本 IP 核有以下优点：

- 本 IP 核含有标准总线接口，而表 2 中的两个 IP 核都只是含有传统的 8051 接口，不便于扩展。
- 本 IP 核能单周期执行一条指令，而表 2 中 DW8051 是 4 周期执行一条指令，而 DR8051 是大概是 2 周期一条指令。因此在相同的时钟频率下，本设计占有速度优势。

4 系统验证

为了较好的验证 8051 MCU IP 核的功能正确性，使用友晶科技提供的 DE2 开发板搭建一个验证

平台。在 FPGA 中使用 SOPC 工具[6]搭建一个 IP 核的验证系统，系统的结构见图 13。Avalon 总线上总共挂了 6 个功能模块。分别是：串口模块、片上存储器、SRAM 接口、PIO 接口、定时器、8051 MCUIP 核。每个模块的功能如下：

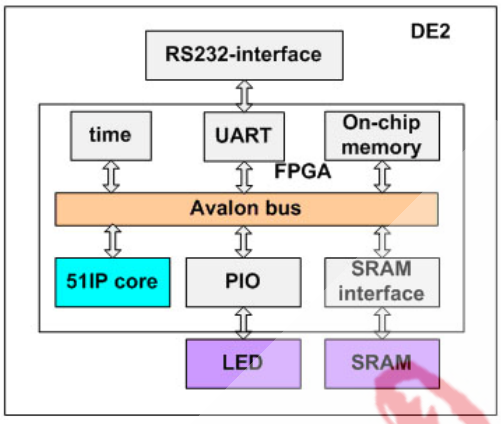

图 13 系统验证结构

- 串口模块：定义的波特率是 9600bps、数据位宽是 8。
- 片上存储器：定义了一个存储容量是 16KB 的空间，存储器的读写数据位宽是 32。
- SRAM 接口：用于连接 DE2 板上的 SRAM 芯片，这个芯片的容量是 512KB。
- PIO 接口：定义了一个 8 位宽的输出并口，用来直接驱动 DE2 板上的 LED，用于显示 IP 核的工作状态。
- 定时器：定义了一个每隔 1ms 产生一个定时中断，中断时输出一个时钟周期的脉冲。
- 51IP 核：被测试的 IP 核。

验证平台的工作流程如下：

1. 搭建硬件系统：将 DE2 和电脑用串口线连接。
2. 程序编译和下载：将待执行的程序编译成机器码，然后将此机器码通过 DE2 控制面板下载到 SRAM 中。这样所需执行的程序已经准备好，见图 14。

图 14 程序的编译和下载

3. 程序的执行：当程序下载到 SRAM 之后，需要将前面搭建的 IP 核验证系统的 FPGA 配置文件通过下载线来完成对 FPGA 的配置。当配置完成后 IP 核即开始运行。如果需要可以通过 DE2 上的 KEY0 按钮来对 IP 核复位。
4. 结果的分析：在测试 IP 核中，会设计一些用来测试 IP 核的存储器和运算功能的程序，这些程序会将结果实时的通过串口发送出去。在电脑上通过串口调试助手来接收这些数据，通过分析数据来判断指令执行的正确性。

下面给出两组代码的验证结果：

1、测试用户 RAM 区

- 汇编指令：

```
;测试用户 RAM 区
  ;并口输出 01H
      MOV 84H, #08H     ;DSL
      MOV   85H, #00H     ;DSH
      MOV   DPTR, #0040H      ;状态寄存器地址
      MOV   A, #01H  ;1 个绿灯亮
      MOVX      @DPTR, A
      MOV   A, #0
      MOV   R0, #0
INIT_USER_RAM:
      MOV   @R0,A
      INC      A
      INC R0
      CJNE   A,#128,INIT_USER_RAM
      MOV   R0,#0
   ;将内部 RAM 区的数据读取出来，通过串口输出
SEND_USER_RAM_TO_UART:
      MOV    A,@R0
      ACALL      WAIT_TRDY ;用于检测串口是否空闲的子函数
      MOVX @DPTR,A     ;发送数据
      INC R0
      CJNE    R0, #128, SEND_USER_RAM_TO_UART
```

- 汇编功能：

向内部 128B 空间的用户 RAM 区依次写入 0~7FH，然后将这些数据读出，通过串口发送到电脑上。

- 测试结果：

串口显示的数据见图 15。其中显示接收到的时候依次是 00~7FH，表明验证结果正确。

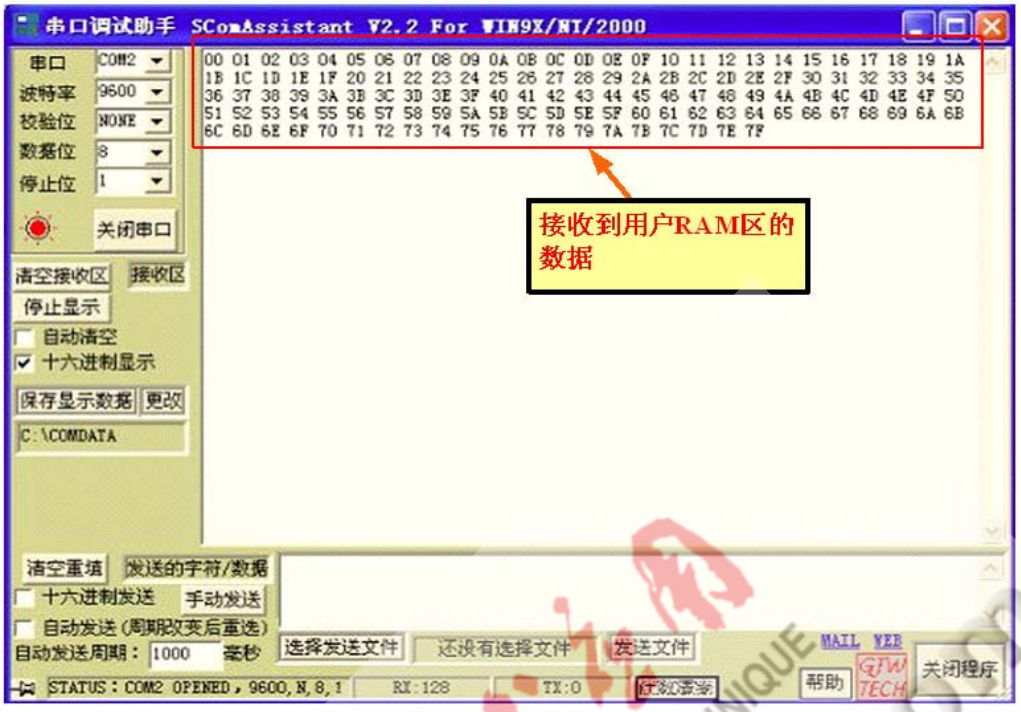

图 15 测试用户 RAM 区的结果

2、测试堆栈区

- 汇编指令：

```
;测试堆栈
   ;并口输出 03H
    MOV     84H,#08H      ;DSL
    MOV     85H,#00H      ;DSH
    MOV     DPTR,#0040H      ;状态寄存器地址
    MOV     A,#03H   ;2 个绿灯亮
    MOVX    @DPTR,A
    MOV     A,#0
INIT_STACK:    MOV      10H,A
    PUSH    10H ;将地址是 10H 的值入栈
    INC A
    CJNE    A,#64,INIT_STACK
    MOV     A,#20H
    MOV     R0,#20H
; 出栈,将值保存在 0H 单元
SEND_STACK_TO_UART:     POP 20H
    MOV     A,20H
    ACALL   WAIT_TRDY
    MOVX    @DPTR,A
    INC R0
    CJNE    R0,#64,SEND_STACK_TO_UART
```

- 汇编功能：

向内部 64B 空间的堆栈区连续入栈 0~3FH 数据，然后将这些数据读出，通过串口发送到电脑上。

- 测试结果：

串口接收的数据见图 16。其中显示接收到的时候依次是 3FH~00H，因为入栈时是 0~3FH，出栈时顺序相反。因此表明验证结果正确。

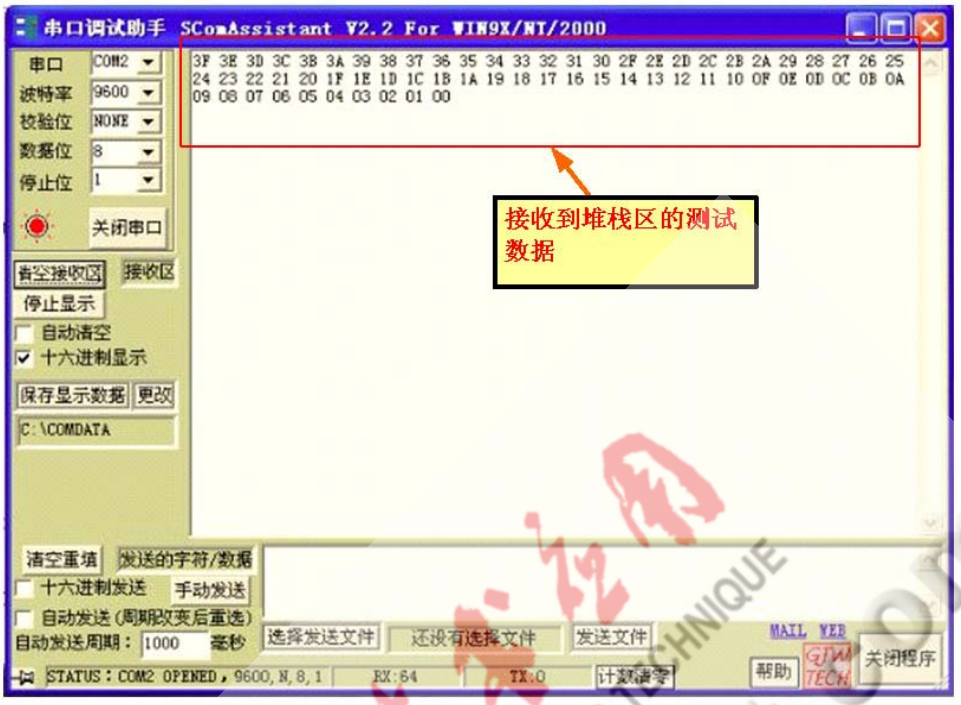

图 16 测试堆栈区的结果

通过上面的部分测试结果可以看出本 IP 核工作正常，能够基本执行各种指令，达到设计要求。

5 总结

本文设计了一款基于 Avalon 总线的 8051MCU IP 核，它具有以下特点：

1. 采用流水线技术、指令映射和预取技术等实现了 IP 核内部的结构优化，使其指令能够在单周期内执行完，并且整个 IP 核的工作速度能够达到 100MHz（CycloneII EP2C35F672C6），这样该 IP 核的峰值处理速度达到 100MIPS。
2. IP 核使用 Avalon 总线接口，能够很方便的使用 SOPC 工具构建 Avalon 总线系统，满足各种应用需求。

参考文献

[1] 何立民. 从 Cygnal C8051F 看 8 位单片机发展之路[J]. 单片机与嵌入式系统应用，2002.5：5-8

[2] DW8051 Datasheet. http://www.keil.com/dd/docs/datashts/synopsys/dw8051_ds.pdf

[3] 胡欣幸. 一种带指令高速缓存的 8051IP[J]. 机电工程，2008 年 10 月，67-70.

[4] Altera Corporation. Avalon Interface Specifications. Oct. 2008，Ver 1.1

[5] 李广弟，朱月秀，王秀山. 单片机基础(修订本)[M]. 北京：北京航空航天大学出版社，2001.7：3-4.

[6] 周博，邱卫东，陈燕，周学功，方茁. 挑战 SOC-基于 NIOS 的 SOPC 设计与实践[M]. 北京：清华大学出版社，2004.7：67