



Imagination大学计划

RVfpga实验7

7段显示屏

1. 简介

本实验将介绍如何扩展RVfpga系统以便与7段显示屏配合使用，还演示了如何修改7段显示屏控制器。Nexys A7 FPGA开发板有8个7段显示屏。我们首先介绍其工作原理（第2部分），然后分析RVfpga系统中包含的8位7段显示屏控制器的高级规范并提供一些基本练习（第3部分和第4部分）。最后分析该控制器的底层实现，执行Verilator仿真并提供其他练习，您将在这些练习中修改和测试该控制器的实现（第5部分和第6部分）。

2. NEXYS A7开发板上的7段显示屏

Nexys A7开发板包含两个4位共阳极7段LED显示屏¹，经配置可用作单个8位7段显示屏（见图1）。八位数字中的每一位都由七个段组成，这七个段以“图形8”模式排列（见图2），每一段都有一个LED。其中的每一段都可以点亮或熄灭，因此通过点亮某些LED段并熄灭其他段，一位数字上可以显示128种模式中的任何一种；具体来说，在这128种模式中，可以显示十进制数字，如图2所示。

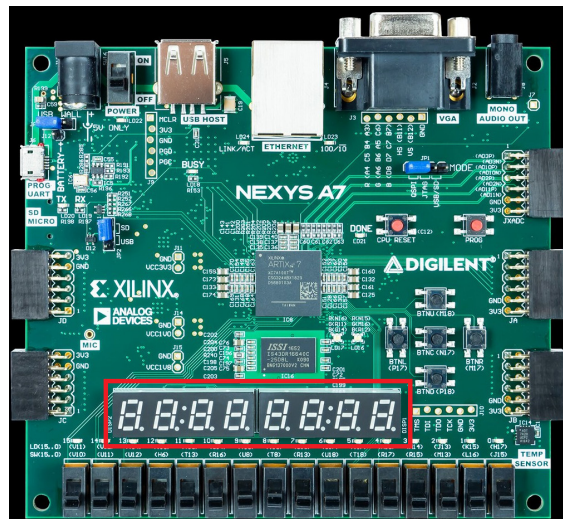


图1. Nexys A7上的8位7段显示屏

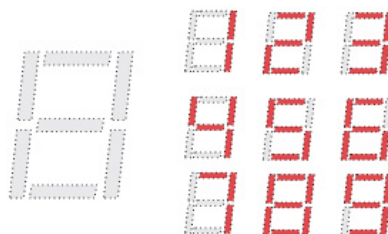


图2. 对应于十进制数字的模式

¹以下地址介绍了本部分中的信息：<https://reference.digilentinc.com/reference/programmable-logic/nexys-a7/reference-manual>

一位数字的LED段标记为A-G，如图3右侧所示。每一位的七个LED的阳极连接到同一个“共阳极”电路节点，但LED阴极保持分离（见图3）。八个共阳极信号用作“数字使能”，每位数字（AN0-AN7）对应一个共阳极信号。所有八位数字的同一个段的阴极连接到CA-CG这七个信号（见图3）。（请注意，第八个信号对应于十进制小数点DP，但本实验中不会使用它。）例如，八位数字的D段的阴极分组到称为CD的同一个电路节点中。这种信号连接方案构成了一个多路复用显示屏，其中阴极信号是所有数字通用的，但它们只能将对应阳极信号置为有效的数字段点亮。所有这些信号在有效时都会被驱动为低电平；因此，要点亮某个段（例如第2位数字的D段），阳极AN2和阴极CD都必须驱动为低电平。

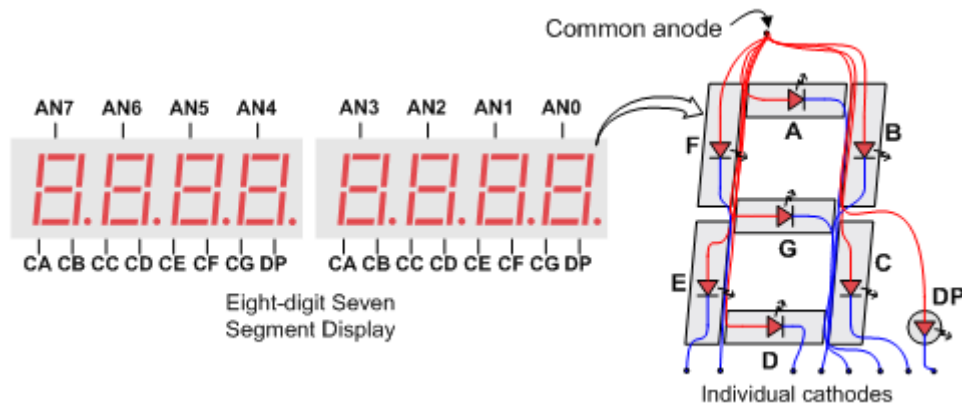


图3. Nexys A7上的8位7段显示屏的连接

扫描显示控制器电路可用于在8位7段显示屏上显示8位数字。该电路以人眼无法检测到的更新速率连续不断地驱动每位数字的阴极；同时，电路一次驱动一个阳极。因此，每位数字仅有八分之一的时间被点亮，但是，由于人眼无法在某位数字再次点亮之前感知到其变暗，因此该位数字看起来是连续被点亮的。

为了使8位数字中的每一位看起来都显得明亮并且持续点亮，应当每隔1到16 ms驱动一次全部8位数字，并且每位数字应点亮1/8的刷新周期（例如，对于16 ms的刷新周期，每位数字应点亮2 ms）。如上文所述，控制器必须以正确的模式将某位数字的阴极驱动为低电平，同时将相应的阳极信号也驱动为低电平。但是，由于Nexys A7使用NPN晶体管将足够的电流驱动到共阳极点，故此阳极使能信号会反转。因此，AN0...7和CA...G/DP信号在有效时被驱动为低电平。

为了说明该过程，假设您要在最右边的两个数字上显示71。控制器电路会在前2 ms将AN0、CB和CC驱动为低电平，从而在最右边的一位数字上显示1，然后，在接下来的2 ms将AN1、CA、CB和CC驱动为低电平，从而在下一个最高有效数字上显示7。如果此过程无限重复，人眼将在最右边的两位数字上看到71。

3. 8位7段显示屏控制器的高级规范

在本部分中，我们首先说明和分析RVfpga系统中使用的8位7段显示屏控制器的高级规范，然后提供使用它的练习。

A. 高级规范

本课程中使用的8位7段显示屏控制器是专为RVfpga系统定制设计的。它包含两个名为 *Enables_Reg* 和 *Digits_Reg* 的寄存器，分别映射到地址0x80001038和0x8000103C（请注意，这两个地址是为系统控制器保留的地址范围内的未使用地址，可以在 <https://github.com/chipsalliance/Cores-SweRVolf> 中查看）。

任务：找到寄存器 *Enables_Reg* 和 *Digits_Reg* 的声明以及为其分配值的位置。8位7段显示屏在以下文件中实现：

`[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf_syscon.v.`

Enables_Reg 是一个8位寄存器，其中的每一位用于确定相应位数字为 **ON**（0）还是 **OFF**（1）。*Digits_Reg* 是一个32位寄存器，其中每个4位组代表要在相应位数字上显示的十六进制值。例如，要在最右边的两位数字上显示71，编程器应为寄存器分配以下值：

- *Enables_Reg* = 0xFC （使能最右边的两位数字）
- *Digits_Reg* = 0x00000071 （值 = 71）

4. 基本练习

练习1. 编写一个RISC-V汇编程序和一个C程序，在7段显示屏最右边的四位数字上显示开关值。

练习2. 编写一个RISC-V汇编程序和一个C程序，在8位7段显示屏上从右到左移动显示字符串“0-1-2-3-4-5-6-7-8”。也就是说，最右边的一位数字上应先显示0。之后，0应向左移动，最右边的一位数字上应显示1，依此类推。

5. 8位7段显示屏控制器：底层实现和仿真

到目前为止，我们仅说明了如何使用8位7段显示屏。在本部分中，我们将介绍其底层实现，并在执行简单汇编示例时通过仿真分析RVfpgaSim。最后，我们将提供修改8位7段显示屏控制器的练习。

A. 8位7段显示屏控制器的底层实现

与以前的通用I/O（General-Purpose I/O, GPIO）实验相似，我们将8位7段显示屏控制器的分析分为三个阶段：

1. 板上SoC和I/O器件之间的连接（图4中的左侧阴影区域）；
2. 新控制器的集成，该新控制器包含在SoC中的SweRVVolFX系统控制器中（图4中的中间阴影区域）；
3. 新控制器与SweRV EH1内核之间的连接（图4中的右侧阴影区域）。

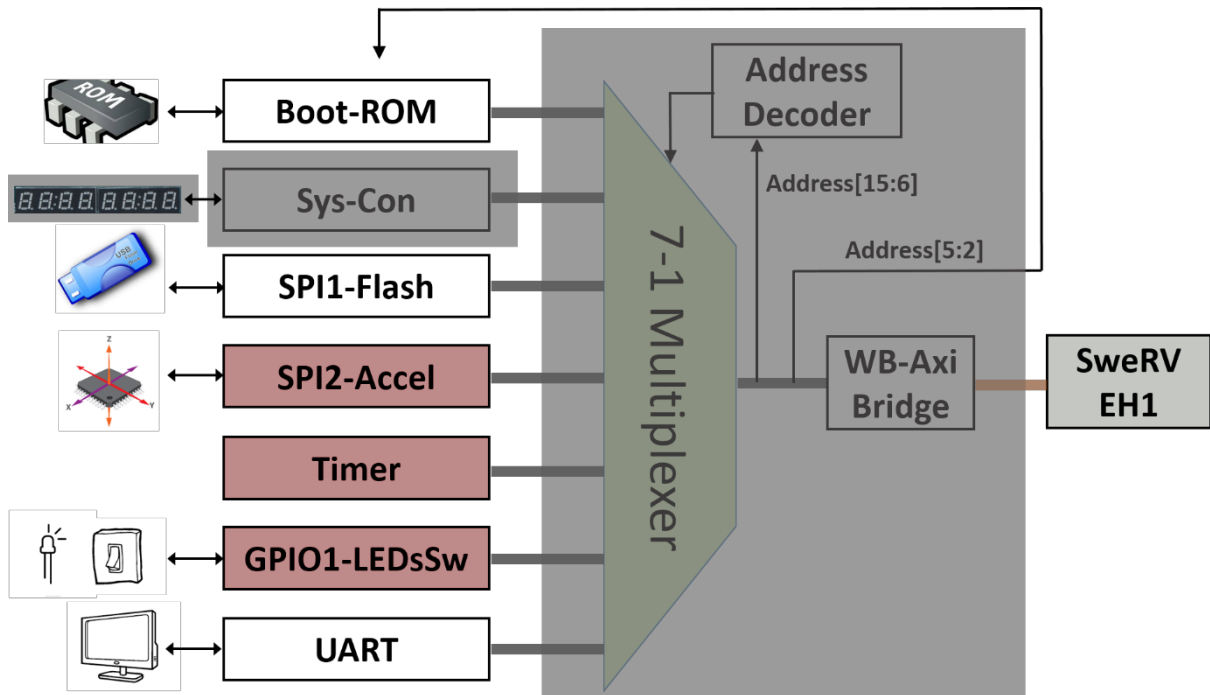


图4. 8位7段显示屏控制器分析（分为3个阶段）

1. LED/开关与SoC的连接

项目的约束文件（`[RVfpgaPath]/RVfpga/src/rvfpganexys.xdc`）定义了输入/输出SoC信号与开发板之间的连接。Nexys A7 FPGA开发板上的每个I/O器件都连接到特定的FPGA引脚。连接八个阳极的信号（见图3）称为 $AN[i]$ （ i 的范围为0-7），连接所有8位数字上相似段的阴极的信号（见图3）称为 CA 、 CB 、 CC 、 CD 、 CE 、 CF 和 CG 。图5给出了定义这些连接的约束文件的片段。

```
60 ##7 segment display
61 set_property -dict { PACKAGE_PIN T10 IOSTANDARD LVCMOS33 } [get_ports { CA }]; #IO_L24N_T3_A00_D16_14 Sch=ca
62 set_property -dict { PACKAGE_PIN R10 IOSTANDARD LVCMOS33 } [get_ports { CB }]; #IO_25_14 Sch=cb
63 set_property -dict { PACKAGE_PIN K16 IOSTANDARD LVCMOS33 } [get_ports { CC }]; #IO_25_15 Sch=cc
64 set_property -dict { PACKAGE_PIN K13 IOSTANDARD LVCMOS33 } [get_ports { CD }]; #IO_L17P_T2_A26_15 Sch=cd
65 set_property -dict { PACKAGE_PIN P15 IOSTANDARD LVCMOS33 } [get_ports { CE }]; #IO_L13P_T2_MRCC_14 Sch=ce
66 set_property -dict { PACKAGE_PIN T11 IOSTANDARD LVCMOS33 } [get_ports { CF }]; #IO_L19P_T3_A10_D26_14 Sch=cf
67 set_property -dict { PACKAGE_PIN L18 IOSTANDARD LVCMOS33 } [get_ports { CG }]; #IO_L4P_T0_D04_14 Sch=cg
68 #set_property -dict { PACKAGE_PIN H15 IOSTANDARD LVCMOS33 } [get_ports { DP }]; #IO_L19N_T3_A21_VREF_15 Sch=dp
69 set_property -dict { PACKAGE_PIN J17 IOSTANDARD LVCMOS33 } [get_ports { AN[0] }]; #IO_L23P_T3_F0E_B_15 Sch=an[0]
70 set_property -dict { PACKAGE_PIN J18 IOSTANDARD LVCMOS33 } [get_ports { AN[1] }]; #IO_L23N_T3_FWE_B_15 Sch=an[1]
71 set_property -dict { PACKAGE_PIN T9 IOSTANDARD LVCMOS33 } [get_ports { AN[2] }]; #IO_L24P_T3_A01_D17_14 Sch=an[2]
72 set_property -dict { PACKAGE_PIN J14 IOSTANDARD LVCMOS33 } [get_ports { AN[3] }]; #IO_L19P_T3_A22_15 Sch=an[3]
73 set_property -dict { PACKAGE_PIN P14 IOSTANDARD LVCMOS33 } [get_ports { AN[4] }]; #IO_L8N_T1_D12_14 Sch=an[4]
74 set_property -dict { PACKAGE_PIN T14 IOSTANDARD LVCMOS33 } [get_ports { AN[5] }]; #IO_L14P_T2_SRCC_14 Sch=an[5]
75 set_property -dict { PACKAGE_PIN K2 IOSTANDARD LVCMOS33 } [get_ports { AN[6] }]; #IO_L23P_T3_35 Sch=an[6]
76 set_property -dict { PACKAGE_PIN U13 IOSTANDARD LVCMOS33 } [get_ports { AN[7] }]; #IO_L23N_T3_A02_D18_14 Sch=an[7]
```

图5. 8位7段显示屏输入的连接（文件`rvfpganexys.xdc`）

任务：跟踪从约束文件到系统控制器模块的这些信号（CA-CG和AN），CA-CG在系统控制器模块中合并为数组*Digits_Bits*。您将需要检查以下文件：

```
[RVfpgaPath]/RVfpga/src/rvfpaganexys.xdc
[RVfpgaPath]/RVfpga/src/rvfpaganexys.sv
[RVfpgaPath]/RVfpga/src/SweRVolfSoC/swervolf_core.v
[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf_syscon.v
```

2. 将8位7段显示屏控制器集成到SoC中

在模块*swervolf_syscon*

（*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf_syscon.v*）的第276-288行，8位7段显示屏控制器经过实例化并集成到SoC中（参见图8）。

```
276 // Eight-Digit 7 Segment Displays
277
278 reg [ 7:0] Enables_Reg;
279 reg [31:0] Digits_Reg;
280
281 SevSegDisplays_Controller SegDispl_Ctr(
282     .clk           (i_clk),
283     .rst_n         (i_rst),
284     .Enables_Reg   (Enables_Reg),
285     .Digits_Reg    (Digits_Reg),
286     .AN            (AN),
287     .Digits_Bits   (Digits_Bits)
288 );
289
290 endmodule
```

图8. 8位7段显示屏控制器实例化（文件：*swervolf_syscon.v*）

除了时钟信号（*i_clk*，重命名为*clk*）和复位信号（*i_rst*，重命名为*rst_n*）外，**SevSegdisplays_Controller**模块还接收两个输入信号（*Enables_Reg*和*Digits_Reg*），即前文所述的两个存储器映射控制寄存器。该模块输出两个信号（*AN*和*Digits_Bits*），这两个信号连接到板上7段显示屏。对于在最右边的两位数字上显示71的示例，**SevSegdisplays_Controller**将会将以下值分配给信号*AN*和*Digits_Bits*：

- 从0 ms到2 ms：信号*AN[0]*为低电平以显示第0位数字（最右边的一位数字）。信号*Digits_Bits[5]*和*Digits_Bits[4]*（对应于CB和CC）也为低电平以在第0位数字（最右边的一位数字）上显示“1”。所有其他信号均为高电平。
- 从2 ms到4 ms：信号*AN[1]*为低电平以显示第1位数字。*Digits_Bits[6]*、*Digits_Bits[5]*和*Digits_Bits[4]*（对应于CA、CB和CC）为高电平以在第1位数字上显示“7”。所有其他信号均为高电平。
- 从4 ms到16 ms：*AN[2]…AN[7]*在2 ms间隔内为高电平，因此它们不显示值。其余数字（第2-7位数字）的段也为高电平。

SevSegdisplays_Controller模块在文件*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf_syscon.v*的第295-366行实现。其中包含以下子单元：

- 两个多路开关用于选择每2 ms发送到*AN*和*Digits_Bits*信号的值。多路开关在模块**SevSegMux**内部实现。
- 为了产生2 ms周期，我们使用文件*counter.sv*和*delta_counter.sv*中提供的**counter**模块，这两个文件均包含在文件夹*[RVfpgaPath]/RVfpga/src/OtherSources/pulp-platform.org__common_cells_1.20.0/src*中。计数器配置为从0计数到2¹⁹，大约每2 ms改变一次的3个最高有效位用作上述两个多路开关的选择信号。

- 译码器在模块**SevenSegDecoder**中实现，用于输出给定4位十六进制值的段值。

任务：详细分析**SevSegdisplays_Controller**模块。下一部分中执行的仿真可以帮助您完成这项任务。如有需要，您还可以使用新信号扩展仿真。

3. 8位7段显示屏控制器与SweRV EH1内核之间的连接

如实验6中所述，设备控制器通过多路开关与SweRV EH1内核连接（见图4）。请记住，7:1多路开关（图9）在文件

[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Interconnect/WishboneInterconnect/wb_intercon.v中实现，该文件在文件

[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Interconnect/WishboneInterconnect/wb_intercon.vh的第104-205行实例化。后一个文件包含在**swervolf_core**模块的第168行，该模块位于：**[RVfpgaPath]/RVfpga/src/SweRVolfSoC/swervolf_core.v**。

多路开关选择要读取或写入哪个外设，根据地址（第110-111行）将CPU（**wb_io_***信号 – 图9的第115-126行）与一个外设的Wishbone总线（图9的第127-138行）连接。例如，如果CPU生成的地址在0x80001000-0x8000103F范围内，则选择系统控制器，从而将信号**wb_io_***与信号**wb_sys_***连接。

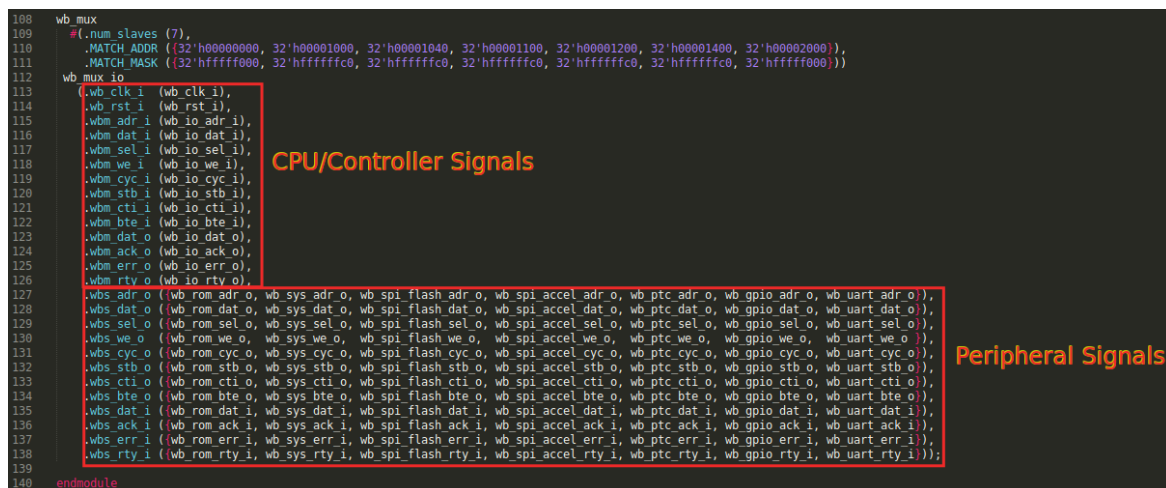


图9. 选择与CPU连接的外设的7-1多路开关（文件：**wb_intercon.v**）

系统控制器中包含的寄存器从CPU写入，具体方法是基于CPU生成的地址（**i_wb_adr**）将这些寄存器与Wishbone总线的数据信号（**i_wb_dat**）直接连接（模块**swervolf_syscon**的第162-228行）。

任务：检查模块**swervolf_syscon**的第162-228行，了解如何在系统控制器中映射地址。重点关注第219-227行（图10），即寄存器**Enables_Reg**和**Digits_Reg**（如前文所述，分配给这两个寄存器的地址分别为0x80001038和0x8000103C）。

```

219      14 : begin
220          if (i_wb_sel[0]) Enables_Reg[7:0]  <= i_wb_dat[7:0];
221      end
222      15 : begin
223          if (i_wb_sel[0]) Digits_Reg[7:0]   <= i_wb_dat[7:0];
224          if (i_wb_sel[1]) Digits_Reg[15:8]  <= i_wb_dat[15:8];
225          if (i_wb_sel[2]) Digits_Reg[23:16] <= i_wb_dat[23:16];
226          if (i_wb_sel[3]) Digits_Reg[31:24] <= i_wb_dat[31:24];
227      end

```

图10. 8位7段显示屏与内核的连接（文件**swervolf_syscon.v**）

B. Verilator仿真

在本部分中，我们使用**RVfpgaSim**在处理器执行驱动该外设的简单示例时检查8位7段显示屏控制器的主要信号。在仿真过程中，我们会在执行图11所示的示例（向最右边的两位数字写入71）时分析信号**AN**和**Digits_Bits**。该程序位于以下位置：

[RVfpgaPath]/RVfpga/Labs/Lab7/71_7SegDispl（也提供了C版本，位置如下：
[RVfpgaPath]/RVfpga/Labs/Lab7/71_7SegDispl_C-Lang）。

```

#define SegEn_ADDR    0x80001038
#define SegDig_ADDR   0x8000103C

.globl main
main:

    li t1, SegEn_ADDR
    li t6, 0xFC
    sb t6, 0(t1)                # Enable the 7SegDisplays

    li t1, SegDig_ADDR
    li t6, 0x71
    sw t6, 0(t1)                # Write the 7SegDisplays

next: beq zero, zero, next

.end

```

图11. 71_7SegDispl.S示例

图12给出了71_7SegDispl.elf程序的反汇编版本，在PlatformIO中编译后的位置如下：

[RVfpgaPath]/RVfpga/Labs/Lab7/71_7SegDispl/.pio/build/swervolf_nexys/firmware.dis

```

00000090 <main>:
90: 80001337      lui      t1,0x80001
94: 03830313      addi     t1,t1,56 # 80001038
98: 0fc00f93      li       t6,252
9c: 01f30023      sb       t6,0(t1)
a0: 80001337      lui      t1,0x80001
a4: 03c30313      addi     t1,t1,60 # 8000103c
a8: 07100f93      li       t6,113
ac: 01f32023      sw       t6,0(t1)

```

```
000000b0 <next>:
    b0: 00000063          beqz    zero,b0 <next>
```

图12. 71_7SegDispl.S示例的反汇编版本

请按照以下步骤运行仿真。（如果您不想运行仿真，则可以直接转到步骤7。）

1. 在这种情况下，出于仅仿真的目的，您必须通过将COUNT_MAX（参见文件[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf_syscon.v的第295行）从20更改为5来缩短时钟周期。否则，查看结果花费的时间将过长。修改COUNT_MAX的值，然后通过执行以下命令重新编译RVfpgaSim（GSG中对此进行了说明）：

```
cd [RVfpgaPath]/RVfpga/verilatorSIM
make clean
make
```

新文件Vrvfpgasim（RVfpgaSim仿真二进制文件）应在目录[RVfpgaPath]/RVfpga/verilatorSIM内生成。

WINDOWS: 如果您使用的是Windows，则必须在Cygwin终端内执行这些命令（有关详细说明，请参见《入门指南》中的第6部分和附录C）。请注意，C: Windows文件夹在Cygwin内的位置为：/cygdrive/c。

MacOS: 有关详细说明，请参见《入门指南》的附录D。

2. 在计算机上打开VSCode/PlatformIO。
3. 在顶部栏上，单击“File”（文件） - “Open Folder...”（打开文件夹...），然后导航至目录[RVfpgaPath]/RVfpga/Labs/Lab7
4. 选择目录71_7SegDispl（不要打开，只需将其选中），然后单击“OK”（确定）。该示例随即将在PlatformIO中打开。
5. 打开文件platformio.ini，检查RVfpgaSim仿真二进制文件的路径是否正确。根据入门指南，其形式应如下所示：

```
board_debug.verilator.binary =
[RVfpgaPath]/RVfpga/verilatorSIM/Vrvfpgasim
```
6. 单击左侧功能区菜单中的PlatformIO图标运行仿真，然后展开“Project Tasks”（项目任务）→ env:swervolf_nexys → “Platform”（平台），并单击“Generate Trace”（生成跟踪）。

文件trace.vcd应已在[RVfpgaPath]/RVfpga/Labs/Lab7/71_7SegDispl/.pio/build/swervolf_nexys内部生成，可以使用GTKWave执行以下命令将其打开：

gtkwave [RVfpgaPath]/RVfpga/Labs/Lab7/71_7SegDispl/.pio/build/swervolf_nexys/trace.vcd

WINDOWS: 已下载的文件夹`gtkwave64`包括一个称为`gtkwave.exe`的应用程序，该应用程序位于`bin`文件夹内。双击该应用程序启动GTKWave。在应用程序的顶部，单击“**File**”（文件）– “**Open New Tab**”（打开新选项卡），然后打开在文件夹`[RVfpgaPath]/RVfpga/Labs/Lab7/71_7SegDispl/.pio/build/swervolf_nexys`中生成的`trace.vcd`文件。

7. 在仿真中包含以下信号（进入参考模块找到每个信号）：
 - rvfpgasim – swervolf – syscon – SegDispl_Ctr
 - ✓ 输入信号：**Enables_Reg**和**Digits_Reg**。
 - ✓ 输出信号：**AN**和**Digits_Bits**。
8. 分析如图13所示的仿真。最初，八个7段显示屏上显示的值均为0（最初所有位数字均使能为**Enables_Reg** = 0）。随后，我们通过将**0xFC**写入**Enables_Reg**（图12中的sb指令）禁止最左边的六位数字，然后将**0x71**写入**Digits_Reg**（图12中的sw指令）向最右边的两位数字写入**71**。对输出信号的影响如下（如图13所示）：
 - 在第一个周期：**AN** = **0xFE**且**Digits_Bits** = **0x4F**，因此最右边的一位数字（第0位数字）上显示**1**。
 - 在第二个周期：**AN** = **0xFD**且**Digits_Bits** = **0x0F**，因此下一位数字（第1位数字）上显示**7**。
 - 在接下来的六个周期：**AN** = **0xFF**且**Digits_Bits** = **0x01**，因此最左边的六位数字熄灭。
 - 此过程随后重复进行。

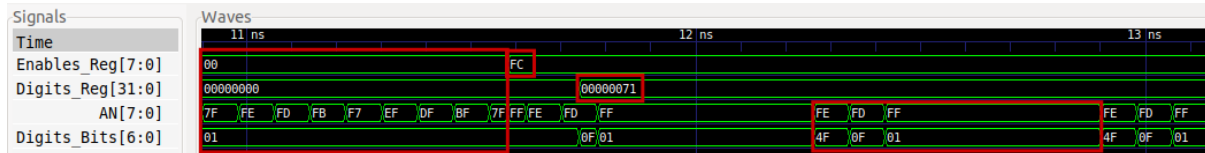


图13. 在8位7段显示屏最右边的两位数字上写入值**71**

9. 在继续操作之前，请不要忘记将**COUNT_MAX**的值恢复为其原始值（**COUNT_MAX** = 20）。

6. 高级练习

练习3. 修改本实验中所说的控制器，以便8位7段显示屏可以显示ON/OFF LED的任意组合。

- 现在不需要使能寄存器，而是需要八个7位寄存器。名称如下：**Segments_Digit0**-**Segments_Digit7**，八个7段显示屏各对应一个名称。其中每个寄存器的每一位用于指示相应段是ON（0）还是OFF（1）。例如，如果第一个寄存器（**Segments_Digit0**）的所有位均为0，则最右边的一位数字的所有段均为ON，而如果第一个寄存器的所有位均为1，则最右边的一位数字的所有段均为OFF。

- 可以将这两个新寄存器映射到我们之前使用的相同地址（先删除之前的两个寄存器 *Enables_Reg* 和 *Digits_Reg*）：
 - Segments_Digit0 \leftrightarrow 地址0x80001038
 - Segments_Digit1 \leftrightarrow 地址0x80001039
 - ...
 - Segments_Digit7 \leftrightarrow 地址0x8000103F
- 请注意，不再需要4-7译码器（模块 **SevenSegDecoder**），因为程序提供的信息已被译码。

练习4. 使用新控制器在8位7段显示屏上显示以下内容：“I SAY HI”。与往常一样，同时实现程序的RISC-V汇编版本和C版本。