

**IMAGINATION大學計劃**

**RVfpga實驗7**

**7段顯示器**

# 簡介

本實驗將介紹如何擴展RVfpga系統以便與7段顯示器配合使用，還示範了如何修改7段顯示器控制器。Nexys A7 FPGA開發板8個7段顯示器。我們首先介紹其工作原理（第2節），然後分析RVfpga系統中包含的8位7段顯示器控制器的高階規格並提供一些基本練習（第3節和第4節）。最後分析該控制器的低階實作，執行Verilator模擬並提供其他練習，您將在這些練習中修改和測試該控制器的實作（第5節和第6節）。

# NEXYS A7開發板上的7段顯示器

# Nexys A7開發板包含兩個4位共陽極7段LED顯示器[[1]](#footnote-1)，經配置可用作單一8位7段顯示器（請參閱圖1）。八位數字中的每一位都由七個段組成，這七個段以「圖形8」模式排列（請參閱圖2），每一段都有一個LED。其中的每一段都可以點亮或熄滅，因此通過點亮某些LED段並熄滅其他段，一位數字上可以顯示128種模式中的任何一種；具體來說，在這128種模式中，可以顯示十進位數字，如圖2所示。

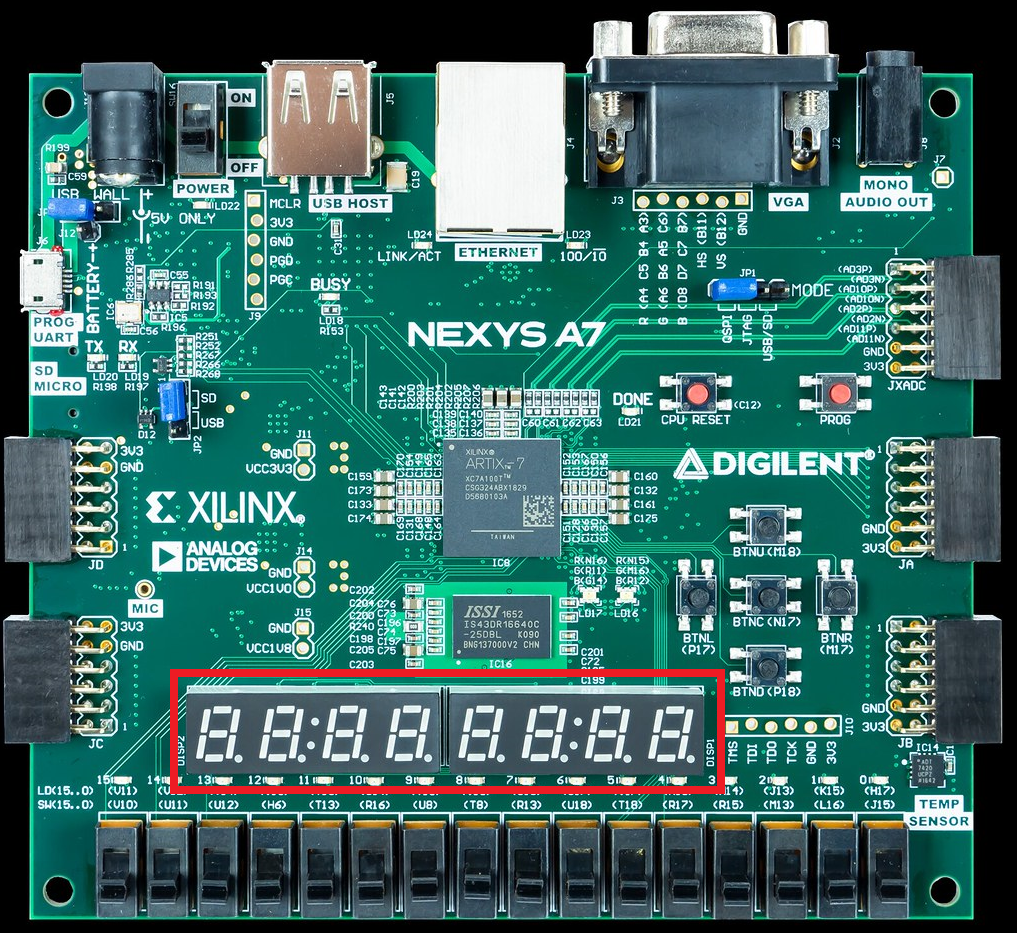


圖. Nexys A7上的8位7段顯示器

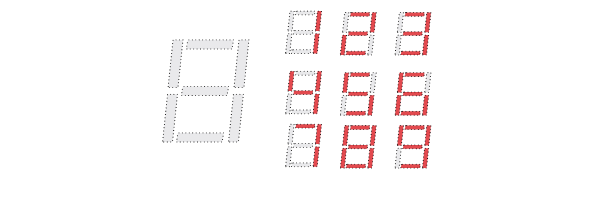


圖. 對應於十進位數字的模式

# 一位數字的LED段標記為*A*-*G*，如圖3右側所示。每一位的七個LED的陽極連接到同一個「共陽極」電路節點，但LED陰極保持分離（請參閱圖3）。八個共陽極訊號用作「數字啟用」，每位數字（*AN0*-*AN7*）對應一個共陽極訊號。所有八位數字的同一個段的陰極連接到*CA*-*CG*這七個訊號（請參閱圖3）。（請注意，第八個訊號對應於十進位小數點*DP*，但本實驗中不會使用它。）例如，八位數字的*D*段的陰極分組到稱為*CD*的同一個電路節點中。這種訊號連接方案構成了一個多工顯示器，其中陰極訊號是所有數字通用的，但它們只能將對應陽極訊號置為有效的數欄位點亮。所有這些訊號在有效時都會被驅動為低電平；因此，要點亮某個段（例如第*2*位數字的*D*段），陽極*AN2*和陰極*CD*都必須驅動為低電平。

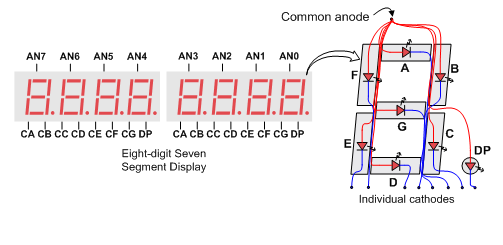


圖. Nexys A7上的8位7段顯示器的連接

掃描顯示控制器電路可用於在8位7段顯示器上顯示8位數字。該電路以人眼無法偵測到的更新速率連續不斷地驅動每位數字的陰極；同時，電路一次驅動一個陽極。因此，每位數字僅有八分之一的時間被點亮，但是，由於人眼無法在某位數字再次點亮之前感知到其變暗，因此該位數字看起來是連續點亮的。

為了使8位數字中的每一位看起來都顯得明亮並且持續點亮，應當每隔1到16 ms驅動一次全部8位數字，並且每位數字應點亮1/8的重新整理週期（例如，對於16 ms的重新整理週期，每位數字應點亮2 ms）。如上文所述，控制器必須以正確的模式將某位數字的陰極驅動為低電平，同時將相應的陽極訊號也驅動為低電平。但是，由於Nexys A7使用NPN晶體管將足夠的電流驅動到共陽極點，故此陽極啟用訊號會反轉。因此，AN0...7和CA...G/DP訊號在有效時被驅動為低電平。

為了說明該程序，假設您要在最右邊的兩個數字上顯示*71*。控制器電路會在前2 ms將*AN0*、*CB*和*CC*驅動為低電平，從而在最右邊的一位數字上顯示*1*，然後，在接下來的2 ms將*AN1*、*CA*、*CB*和*CC*驅動為低電平，從而在下一個最高有效數字上顯示*7*。如果此程序無限重複，人眼將在最右邊的兩位數字上看到*71*。

# 8位7段顯示器控制器的高階規格

在本節中，我們首先說明和分析RVfpga系統中使用的8位7段顯示器控制器的高階規格，然後提供使用它的練習。

1. **高階規格**

本課程中使用的8位元7段顯示器控制器是專為RVfpga系統所量身設計的。它包含兩個名為*Enables\_Reg*和*Digits\_Reg*的暫存器，分別映射到位址0x80001038和0x8000103C（請注意，這兩個位址是為系統控制器保留的位址範圍內的未使用位址，可以在<https://github.com/chipsalliance/Cores-SweRVolf>中檢視）。

**任務：**找到暫存器*Enables\_Reg*和*Digits\_Reg*的宣告以及為其分配值的位置。8位7段顯示器在以下檔案中實作：*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf\_syscon.v*。

*Enables\_Reg*是一個8位元暫存器，其中的每一位元用於確定相應位數字為*ON*（0）還是*OFF*（1）。*Digits\_Reg*是一個32位元暫存器，其中每個4位元組代表要在相應位數字上顯示的十六進位值。例如，要在最右邊的兩位數字上顯示*71*，程式設計師應為暫存器分配以下值：

* *Enables\_Reg = 0xFC* （啟用最右邊的兩位數字）
* *Digits\_Reg = 0x00000071* （值 = 71）

# 基本練習

# 練習1. 編寫一個RISC-V組合語言程式和一個C程式，在7段顯示器最右邊的四位數字上顯示開關值。

# 練習2. 編寫一個RISC-V組合語言程式和一個C程式，在8位7段顯示器上從右到左移動顯示字串「0-1-2-3-4-5-6-7-8」。也就是說，最右邊的一位數字上應先顯示0。之後，0應向左移動，最右邊的一位數字上應顯示1，依此類推。

# 8位7段顯示器控制器：低階實作和模擬

# 到目前為止，我們僅說明了如何使用8位7段顯示器。在本節中，我們將介紹其低階實作，並在執行簡單組合語言程式碼範例時通過模擬分析RVfpgaSim。最後，我們將提供修改8位7段顯示器控制器的練習。

1. **8位7段顯示器控制器的低階實作**

與以前的通用I/O（General-Purpose I/O，GPIO）實驗相似，我們將8位7段顯示器控制器的分析分為三個階段：

1. 開發板上SoC和I/O裝置之間的連接（圖4中的左側陰影區域）；
2. 新控制器的整合，該新控制器包含在SoC中的SweRVolfX系統控制器中（圖4中的中間陰影區域）；
3. 新控制器與SweRV EH1核心之間的連接（圖4中的右側陰影區域）。



圖. 8位7段顯示器控制器分析（分為3個階段）

1. **LED/開關與SoC的連接**

專案的限制檔（*[RVfpgaPath]/RVfpga/src/rvfpganexys.xdc*）定義了輸入/輸出SoC訊號與開發板之間的連接。Nexys A7 FPGA開發板上的每個I/O裝置都連接到特定的FPGA引腳。連接八個陽極的訊號（請參閱圖3）稱為*AN[i]*（*i*的範圍為0-7），連接所有8位數字上相似段的陰極的訊號（請參閱圖3）稱為*CA、CB、CC、CD、CE、CF*和*CG*。圖5顯示了定義這些連接的限制檔片段。

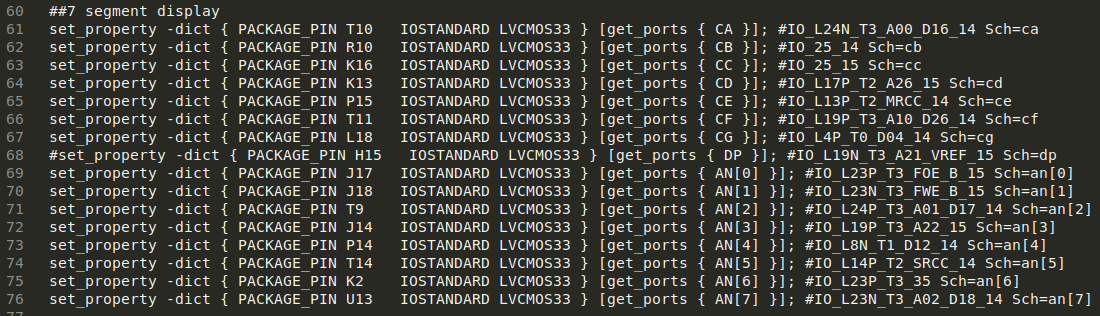


圖. 8位元7段顯示器輸入的連接（檔案*rvfpganexys.xdc*）

在系統頂端模組（模組**rvfpganexys**，在*[RVfpgaPath]/RVfpga/src/rvfpganexys.sv*檔中實作）的第50-51行中，可以找到8位7段顯示器連接到SoC的*AN[i]*和*CA*…*CG*的輸入訊號（圖6的左側部分）；而在該模組的末尾（圖6的右側部分），可以找到該顯示器與**swervolf\_core**模組的連接（請注意，*CA…CG*訊號在該模組中重新命名為*Digits\_Bits[6:0]*）。

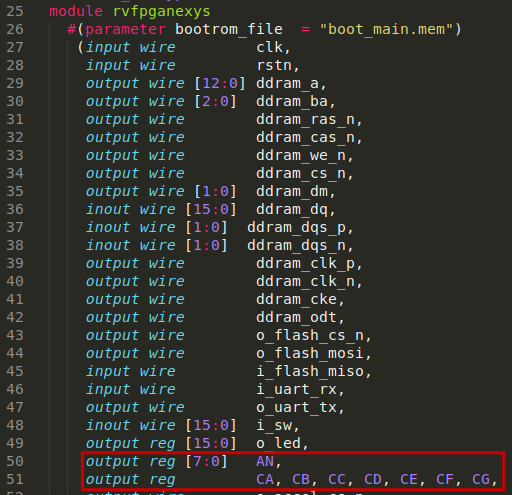
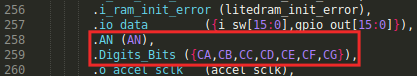
 

圖. 8位7段顯示器與SoC的連接（檔案：*rvfpganexys.sv*）。

最後，兩個訊號從**swervolf\_core**模組插入系統控制器模組（**swervolf\_syscon**）（請參閱圖7），其中實作了8位7段顯示器控制器。

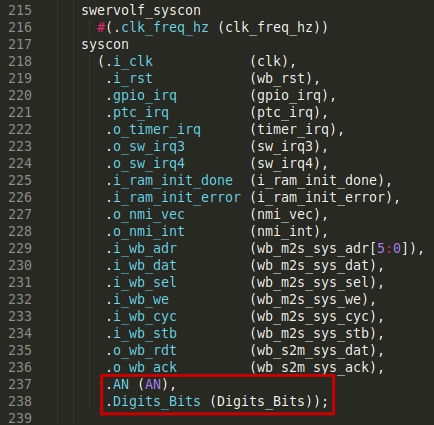


圖. 8位7段顯示器與系統控制器的連接（檔案：*swervolf\_core.v*）。

**任務：**追蹤從限制檔到系統控制器模組的這些訊號（*CA-CG*和*AN*），*CA*-*CG*在系統控制器模組中合併為陣列*Digits\_Bits*。您將需要檢查以下檔案：

*[RVfpgaPath]/RVfpga/src/rvfpganexys.xdc*

*[RVfpgaPath]/RVfpga/src/rvfpganexys.sv*

*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/swervolf\_core.v*

*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf\_syscon.v*

RI

1. **將8位7段顯示器控制器整合到SoC中**

在模組**swervolf\_syscon**（[RVfpgaPath]/*RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf\_syscon.v*）的第276-288行，8位7段顯示器控制器經過實例化並整合到SoC中（請參閱圖8）。

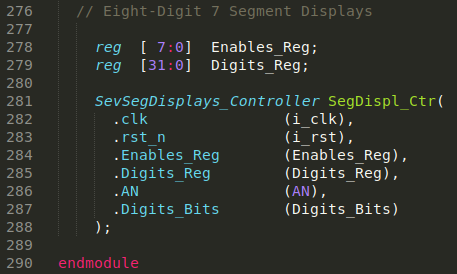


圖. 8位7段顯示器控制器實例化（檔案：*swervolf\_syscon.v*）

除了時鐘訊號（*i\_clk*，重新命名為*clk*）和重設訊號（*i\_rst*，重新命名為*rst\_n*）外，**SevSegdisplays\_Controller**模組還接收兩個輸入訊號（*Enables\_Reg*和*Digits\_Reg*），即前文所述的兩個記憶體映射控制暫存器。該模組輸出兩個訊號（*AN*和*Digits\_Bits*），這兩個訊號連接到開發板上7段顯示器。對於在最右邊的兩位數字上顯示*71*的範例，**SevSegdisplays\_Controller**會將以下值分配給訊號*AN*和*Digits\_Bits*：

* 從0 ms到2 ms：訊號*AN[0]*為低電平以顯示第0位數字（最右邊的一位數字）。訊號*Digits\_Bits[5]*和*Digits\_Bits[4]*(對應於*CB*和*CC*)也為低電平以在第0位數字（最右邊的一位數字）上顯示「1」。所有其他訊號均為高電平。
* 從2 ms到4 ms：訊號*AN[1]*為低電平以顯示第1位數字。*Digits\_Bits[6]、Digits\_Bits[5]*和*Digits\_Bits[4]*（對應於*CA、CB*和*CC*）為高電平以在第1位數字上顯示「*7」*。所有其他訊號均為高電平。
* 從4 ms到16 ms：*AN[2]*…*AN[7]*在2 ms間隔內為高電平，因此它們不顯示值。其餘數字（第2-7位數字）的段也為高電平。

**SevSegdisplays\_Controller**模組在檔案*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf\_syscon.v*的第295-366行實作。其中包含以下子單元：

* 兩個多工器用於選擇每2 ms傳送到*AN*和*Digits\_Bits*訊號的值。多工器在模組**SevSegMux**內部實作。
* 為了產生2 ms週期，我們使用檔案*counter.sv*和*delta\_counter.sv*中提供的**counter**模組，這兩個檔案均包含在資料夾*[RVfpgaPath]/RVfpga/src/OtherSources/pulp-platform.org\_\_common\_cells\_1.20.0/src*中。計數器配置為從0計數到219，將大約每2 ms改變一次的3個最高有效位元，用作上述兩個多工器的選擇訊號。
* 譯碼器在模組**SevenSegDecoder**中實作，用於輸出給定4位元十六進位值的段值。

**任務：**詳細分析**SevSegdisplays\_Controller**模組。下一節執行的模擬可以幫助您完成這項任務。如有需要，您還可以使用新訊號擴展模擬。

1. **8位7段顯示器控制器與SweRV EH1核心之間的連接**

如實驗6中所述，裝置控制器通過多工器與SweRV EH1核心連接（請參閱圖4）。請記住，7:1多工器（圖9）在*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Interconnect/WishboneInterconnect/wb\_intercon.v*檔中實作，該檔案在*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Interconnect/WishboneInterconnect/wb\_intercon.vh*檔的第104-205行實例化。後一個檔案包含在**swervolf\_core**模組的第168行，該模組位於：*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/swervolf\_core.v*。

多工器選擇要讀取或寫入哪個週邊設備，根據位址（第110-111行）將CPU（*wb\_io\_\**訊號 – 圖9的第115-126行）與一個週邊設備的Wishbone匯流排（圖9的第127-138行）連接。例如，如果CPU產生的位址在0x80001000-0x8000103F範圍內，則選擇系統控制器，從而將訊號*wb\_io\_\**與訊號*wb\_sys\_\**連接。



圖. 選擇與CPU連接的週邊設備的7-1多工器（檔案：*wb\_intercon.v*）

系統控制器中包含的暫存器是從CPU寫入，具體方法是基於CPU產生的位址（*i\_wb\_adr*），將這些暫存器與Wishbone匯流排的資料訊號（*i\_wb\_dat*）直接連接（模組**swervolf\_syscon**的第162-228行）。

**任務：**檢查模組**swervolf\_syscon**的第162-228行，瞭解如何在系統控制器中映射位址。重點關注第219-227行（圖10），即暫存器*Enables\_Reg*和*Digits\_Reg*（如前文所述，分配給這兩個暫存器的位址分別為0x80001038和0x8000103C）。

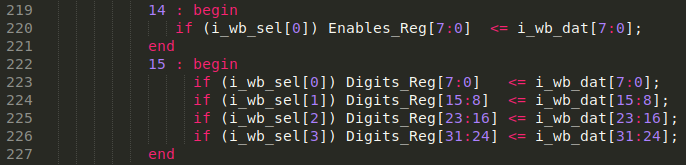


圖. 8位7段顯示器與核心的連接（*swervolf\_syscon.v*檔）

1. **Verilator模擬**

在節中，我們使用**RVfpgaSim**在處理器執行驅動該週邊設備的簡單範例時，檢查8位7段顯示器控制器的主要訊號。在模擬程序中，我們會在執行圖11所示的範例（向最右邊的兩位數字寫入71）時分析訊號*AN*和*Digits\_Bits*。該程式位於以下位置：[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 |

圖. 71\_7SegDispl.S範例

圖12顯示了71\_7SegDispl.elf程式的反組合語言程式碼版本，在PlatformIO中編譯後的位置如下：[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> |

圖12. 71\_7SegDispl.S範例的反組合語言程式碼版本

# 請按照以下步驟執行模擬。（如果您不想執行模擬，則可以直接移至步驟7。）

1. 在這種情況下，出於僅模擬的目的，您必須透過將COUNT\_MAX（參見檔案*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf\_syscon.v*的第295行）從20變更為5來縮短時鐘週期。否則，檢視結果花費的時間將過長。修改COUNT\_MAX的值，然後通過執行以下命令重新編譯RVfpgaSim（GSG中對此進行了說明）：

cd [RVfpgaPath]/RVfpga/verilatorSIM

make clean

make

新檔案*Vrvfpgasim*（RVfpgaSim模擬二進位檔）應在目錄*[RVfpgaPath]/RVfpga/verilatorSIM*內產生。

**WINDOWS：**如果您使用的是Windows，則必須在Cygwin終端機內執行這些命令（有關詳細說明，請參閱《入門指南》中的第6節和附錄C）。請注意，*C:* Windows資料夾位於Cygwin中的以下位置：*/cygdrive/c*。

**MacOS：**有關詳細說明，請參閱《入門指南》的附錄D。

1. 在電腦上開啟VSCode/PlatformIO。
2. 在頂端列上，按一下「*File*」（檔案）→「*Open Folder...*」（開啟資料夾*...*），然後導覽至目錄*[RVfpgaPath]/RVfpga/Labs/Lab7*
3. 選擇目錄*71\_7SegDispl*（不要開啟，只需選擇它），然後按一下「OK」（確定）。該範例將在PlatformIO中開啟。
4. 開啟檔案*platformio.ini*，檢查RVfpgaSim模擬二進位檔案的路徑是否正確。根據入門指南，其形式應如下所示：

board\_debug.verilator.binary = [RVfpgaPath]/RVfpga/verilatorSIM/Vrvfpgasim

1. 按一下左側功能表功能區中的PlatformIO圖示執行模擬，然後展開「Project Tasks」（專案任務）→ env:swervolf\_nexys →「Platform」（平台），並按一下「Generate Trace」（產生軌跡）。

*trace.vcd*檔應已在[RVfpgaPath]/RVfpga/Labs/Lab7/71\_7SegDispl/*.pio/build/swervolf\_nexys*內部產生，可以使用*GTKWave*執行以下命令將其開啟：

gtkwave [RVfpgaPath]/RVfpga/Labs/Lab7/71\_7SegDispl/.pio/build/swervolf\_nexys/trace.vcd

**WINDOWS：**已下載的資料夾*gtkwave64*包括一個稱作*gtkwave.exe*的應用程式，該應用程式位於*bin*資料夾內。按兩下該應用程式啟動GTKWave。在應用程式頂端，按一下**「File」（檔案）–「Open New Tab」（開啟新索引標籤）**，然後開啟在資料夾*[RVfpgaPath]/RVfpga/Labs/Lab7/71\_7SegDispl/.pio/build/swervolf\_nexys*中產生的trace.vcd檔。

1. 在模擬中包含以下訊號（進入參考模組找到每個訊號）：

* rvfpgasim – swervolf – syscon – SegDispl\_Ctr
  + 輸入訊號：***Enables\_Reg***和***Digits\_Reg***。
  + 輸出訊號：***AN***和***Digits\_Bits***。

1. 分析如圖13所示的模擬。最初，八個7段顯示器上顯示的值均為0（最初所有位數字均啟用為*Enables\_Reg* = 0）。隨後，我們透過將*0xFC*寫入*Enables\_Reg*（圖12中的sb指令）停用最左邊的六位數字，然後透過將*0x71*寫入*Digits\_Reg*（圖12中的sw指令）向最右邊的兩位數字寫入*71*。對輸出訊號的影響如下（如圖13所示）：
   * 在第一段期間：*AN = 0xFE*且*Digits\_Bits = 0x4F*，因此最右邊的一位數字（第0位數字）上顯示*1*。
   * 在第二段期間：*AN = 0xFD*且*Digits\_Bits = 0x0F*，因此下一位數字（第1位數字）上顯示*7*。
   * 在接下來的六段期間：*AN = 0xFF*且*Digits\_Bits = 0x01*，因此最左邊的六位數字熄滅。
   * 此程序隨後重複進行。

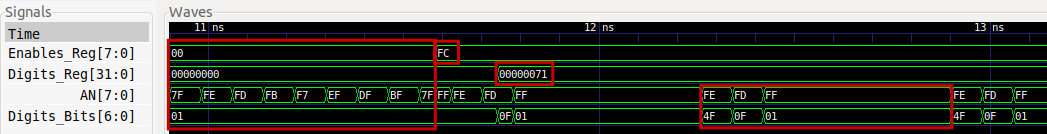


圖. 在8位7段顯示器最右邊的兩位數字上寫入值71

1. 在繼續操作之前，請不要忘記將COUNT\_MAX的值恢復為其原始值（COUNT\_MAX = 20）。

# 進階練習

# 練習3. 修改本實驗中所述的控制器，以便8位7段顯示器可以顯示ON/OFF LED的任意組合。

* 現在不需要啟用暫存器，而是需要八個7位元暫存器。名稱如下：Segments\_Digit0-Segments\_Digit7，八個7段顯示器各對應一個名稱。其中每個暫存器的每個位元用於指示相應段是ON（0）還是OFF（1）。例如，如果第一個暫存器（Segments\_Digit0）的所有位元均為0，則最右邊的一位數字的所有段均為ON；而如果第一個暫存器的所有位元均為1，則最右邊的一位數字的所有段均為OFF。
* 可以將這兩個新暫存器映射到我們之前使用的相同位址（先刪除之前的兩個暫存器*Enables\_Reg*和*Digits\_Reg*）：
  + Segments\_Digit0 🡨🡪 位址0x80001038
  + Segments\_Digit1 🡨🡪 位址0x80001039
  + …
  + Segments\_Digit7 🡨🡪 位址0x8000103F
* 請注意，不再需要4-7譯碼器（模組**SevenSegDecoder**），因為程式提供的資訊已被譯碼。

# 練習4. 使用新控制器在8位7段顯示器上顯示以下內容：「I SAY HI」。與往常一樣，同時實作程式的RISC-V組合語言程式碼版本和C版本。

1. 以下位址介紹了本部分中的資訊：  
   <https://reference.digilentinc.com/reference/programmable-logic/nexys-a7/reference-manual> [↑](#footnote-ref-1)