

**THE IMAGINATION UNIVERSITY PROGRAMME**

**RVfpga Lab 7**

**7 세그먼트 디스플레이**

# 소개

이 LAB에서는 RVfpga시스템의 7 세그먼트 디스플레이에서 작동하도록 확장된 방법과 7 세그먼트 디스플레이 컨트롤러를 수정하는 방법을 보여줍니다. Nexys A7 FPGA 보드에는 8 자리 7 세그먼트 디스플레이가 있습니다. 먼저 작동 방식을 설명하고 (섹션 2) RVfpga시스템 포함된 8 자리 7 세그먼트 디스플레이 컨트롤러의 고급 사양을 분석하고 몇 가지 기본적인 연습을 제공합니다 (섹션 3 및 4). 마지막으로 이 컨트롤러의 초급 수준의 구현을 분석하고 Verilator 시뮬레이션을 수행하며 컨트롤러 구현을 수정하고 실험할 추가 연습을 제공합니다 (섹션 5 및 6).

# NEXYS A7 보드의 7 세그먼트 디스플레이

# Nexys A7 보드에는 단일 8 자리 7 세그먼트 디스플레이로 작동하도록 구성된 2 개의 4 자리 공통 양극 7 세그먼트 LED 디스플레이[[1]](#footnote-1)가 포함되어 있습니다 (그림 1 참조). 8 자리는 각각 "그림 8" 패턴 (그림 2 참조)으로 배열된 7 개의 세그먼트로 구성되며 각 세그먼트에 대한 LED가 있습니다. 이러한 각 세그먼트는 켜거나 끌 수 있으므로 특정 LED 세그먼트를 비추고 나머지는 어둡게 두어 128 패턴 중 하나를 숫자에 표시할 수 있습니다. 특히 이 128 개의 패턴 중 10 진수는 그림 2와 같이 표시할 수 있습니다.

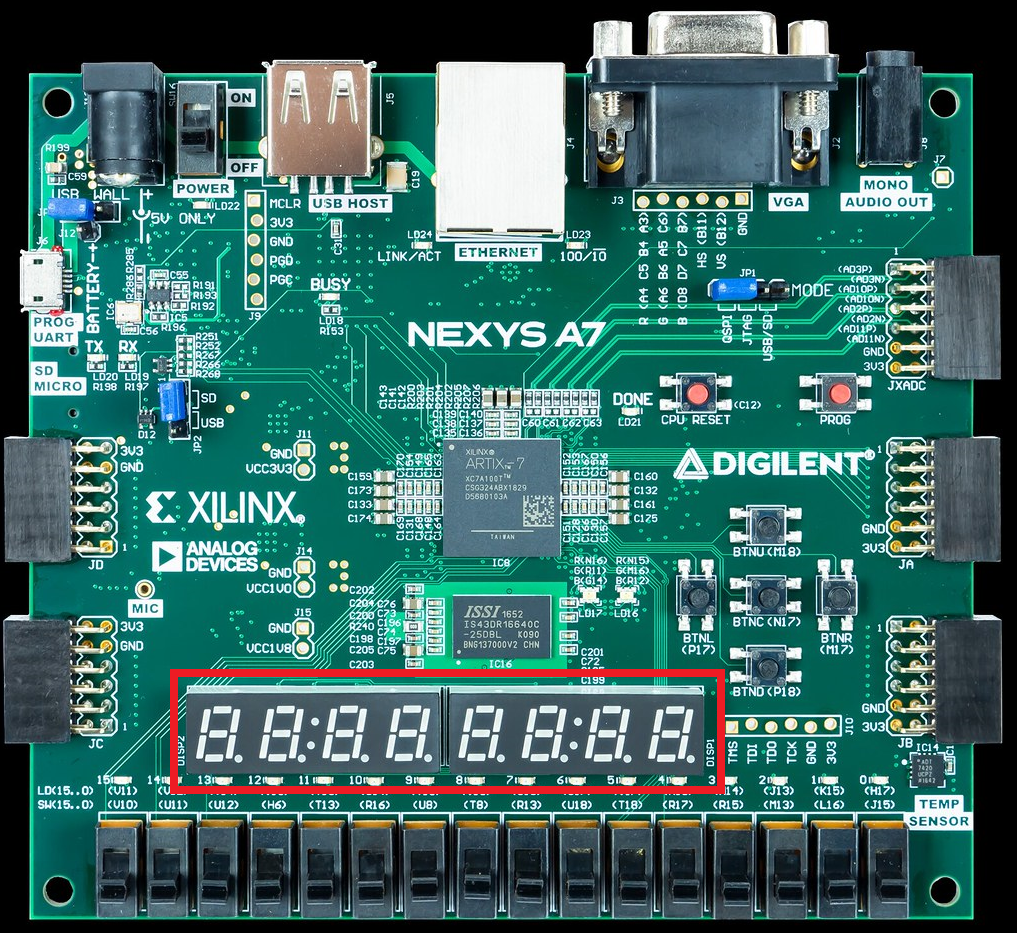


그림 1. Nexys A7의 8 자리 7 세그먼트 디스플레이

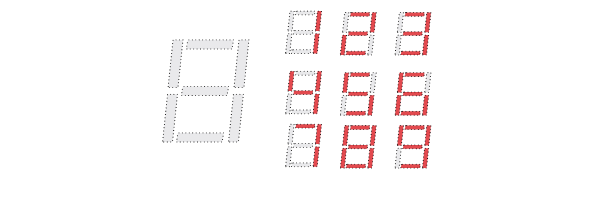


그림 2. 10 진수에 해당하는 패턴

# 한 자리 LED 세그먼트는 그림 3의 오른쪽에 표시된 것처럼 *A-G*로 표시되어 있습니다. 한 자리에 대한 7 개 LED의 양극은 하나의 "공통 양극"회로 노드로 함께 연결되지만 LED 음극은 별도로 유지됩니다 (참조: 그림 3). 8 개의 공통 양극 신호 (각 숫자 (*AN0-AN7*) 당 하나씩)는 "디지트 활성화"역할을 합니다. 8 자리 모두에서 동일한 세그먼트의 음극은 7 개의 신호인 *CA-CG*로(그림 3 참조) 연결됩니다. (소수점 *DP*에 대해 여덟 번째 신호가 존재하지만 이 LAB에서는 사용하지 않습니다.) 예를 들어, 8 자리 숫자에서 세그먼트 *D*의 음극은 *CD*라는 단일 회로 노드로 함께 그룹화 됩니다. 이 신호 연결 방식은 음극 신호가 모든 숫자에 공통적인 다중 디스플레이를 생성하지만, 해당 양극 신호가 표시되는 숫자의 세그먼트만 조명할 수 있습니다. 이 모든 신호는 활성화되면 low로 구동됩니다. 따라서 세그먼트 (예: 숫자 2의 세그먼트 *D*)를 조명하려면 양극 *AN2*와 음극 *CD*를 모두 low로 구동해야 합니다.

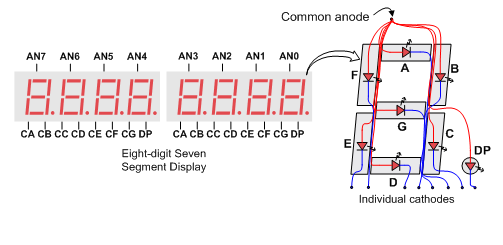


그림 3. Nexys A7의 8자리 7 세그먼트 디스플레이 연결

디스플레이 컨트롤러 회로를 사용하여 7 세그먼트 디스플레이에 8 숫자를 표시 할 수 있습니다. 이 회로는 사람의 눈으로 감지할 수 있는 것보다 더 빠른 업데이트 속도로 반복되는 연속 연속으로 각 숫자의 패턴으로 음극을 구동합니다. 동시에 회로는 양극을 한 번에 하나씩 구동합니다. 따라서 각 숫자는 8 분의 1에 불과하지만 다시 밝아지기 전에는 숫자가 어두워지는 것을 눈으로 인식할 수 없기 때문에 숫자가 계속 켜져 있는 것처럼 보입니다.

8 자리가 밝고 지속적으로 밝게 표시 되려면 8 자리 모두 1 ~ 16ms마다 한 번씩 구동해야 하며 각 자리는 새로 업데이트 주기의 1/8 동안 켜집니다 (예: 16ms 새로 업데이트 주기의 경우 숫자는 2ms 동안 켜집니다). 위에서 설명한 것처럼 컨트롤러는 해당 양극 신호도 low로 구동되는 동안 올바른 패턴으로 디지트 low의 음극을 구동해야 합니다. 그러나 Nexys A7은 NPN 트랜지스터를 사용하여 공통 양극 지점으로 충분한 전류를 구동하기 때문에 양극 활성화가 반전됩니다. 따라서 AN0~7 및 CA~G / DP 신호는 모두 활성 상태일 때 low로 구동됩니다.

프로세스를 설명하기 위해 맨 오른쪽 두 자리에 *71*을 표시한다고 가정합니다. 컨트롤러 회로는 처음 2ms 동안 *AN0, CB* 및 *CC*를 low로 구동하므로 맨 오른쪽 자리에 1이 표시됩니다. 이후 다음 2ms 동안 회로는 *AN1, CA, CB* 및 *CC*를 low로 구동하여 다음 최상위 숫자에 7을 표시합니다. 이 과정이 무한히 반복되면 사람의 눈은 맨 오른쪽 두 자리에 숫자 *71*이 표시됩니다.

# 8-DIGIT 7-SEGMENT 디스플레이 컨트롤러의 높은 수준의 사양

이 섹션에서는 먼저 RVfpga시스템에서 사용되는 8 자리 7 세그먼트 디스플레이 컨트롤러의 고급 사양을 설명하고 분석한 다음, 이를 사용하기 위한 연습을 제공합니다.

1. **고급 사양**

이 과정에서 사용되는 7 세그먼트 디스플레이 컨트롤러는 RVfpga 시스템용으로 맞춤 설계 되었습니다. 여기에는 각각 주소 0x80001038 및 0x8000103C에 매핑된 *Enables\_Reg* 및 *Digits\_Reg*라는 두 개의 레지스터가 포함되어 있습니다 (이러한 주소는 시스템 컨트롤러 용으로 예약된 주소 범위 내에서 사용되지 않는 주소이며 <https://github.com/chipsalliance/Cores-SweRVolf>/에서 볼 수 있음).

**작업:** 레지스터 Enables\_Reg 및 Digits\_Reg의 선언과 값이 할당된 위치를 찾습니다. 세그먼트 디스플레이는 파일로 구현됩니다. *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf\_syscon.v*

*Enables\_Reg*는 각 비트가 해당 숫자가 *ON* (0)인지 *OFF* (1)인지 결정하는 8 비트 레지스터입니다. *Digits\_Reg*는 각 4 비트 그룹이 해당 숫자에 표시할 16 진수 값을 나타내는 32 비트 레지스터입니다. 예를 들어, 가장 오른쪽 두 자리에 *71*을 표시하려면 프로그래머는 레지스터에 다음 값을 할당합니다.

* *Enables\_Reg = 0xFC* (맨 오른쪽 두 자리 활성화)
* *Digits\_Reg = 0x00000071* (값 = 71)

# 기본 실습

# 연습 1. RISC-V 어셈블리 프로그램과 7 세그먼트 디스플레이의 맨 오른쪽 네 자리에 있는 스위치의 값을 보여주는 C 프로그램을 작성하십시오.

# 연습 2. RISC-V 어셈블리 프로그램과 7세그먼트의 오른쪽에서 왼쪽으로 이동하는“0-1-2-3-4-5-6-7-8”문자열을 보여주는 C 프로그램을 작성합니다. 즉, 가장 오른쪽 자리에 0이 먼저 표시되어야 합니다. 그런 다음 왼쪽으로 이동하고 맨 오른쪽 자리에 1이 표시됩니다.

# 8-DIGIT 7-SEGMENT 디스플레이 컨트롤러: 초급 수준의 구현, 시뮬레이션

# 지금까지 7 세그먼트 디스플레이 사용하는 방법을 살펴 보았습니다. 이 섹션에서는 간단한 어셈블리 예제를 실행할 때 초급 수준 구현을 설명하고 시뮬레이션에서 RVfpgaSim를 분석합니다. 마지막으로 7 세그먼트 디스플레이 컨트롤러를 수정하는 연습을 제공합니다.

1. **7 세그먼트 디스플레이 컨트롤러의 초급 수준 구현**

이전의 GPIO (범용 I/O) 실험실과 유사하게 7 세그먼트 디스플레이 컨트롤러의 분석을 세 단계로 나눕니다.

1. 보드의 SoC와 I/O 장치 간의 연결 (그림 4의 왼쪽 그림자 영역)
2. SoC에 포함된 SweRVolfX시스템 컨트롤러 내부에 포함된 새 컨트롤러의 통합 (그림 4의 중간 그림자 영역)
3. 새 컨트롤러와 SweRV EH1 Core 간의 연결 (그림 4의 오른쪽 그림자 영역).



그림 4. 8 자리 7 세그먼트 디스플레이 컨트롤러 분석 3 단계

1. **LED / 스위치를 SoC에 연결**

프로젝트의 constraints file은 (*[RVfpgaPath]/RVfpga/src/RVfpganexys.xdc*) 입력/출력 SoC 신호와 보드 간의 연결을 정의 합니다. Nexys A7 FPGA 보드의 각 I/O 장치는 특정 FPGA 핀에 연결됩니다. 8 개의 양극 (그림 3 참조)을 연결하는 신호를 *AN[i]* (*i* 범위 0-7)라고 하고 8 자리 모두에서 유사한 세그먼트의 음극을 연결하는 신호 (그림 3 참조)를 *CA*라고합니다. , *CB, CC, CD, CE, CF* 및 *CG*. 그림 5는 이러한 연결이 정의된 제약 조건 파일의 일부분을 보여줍니다.

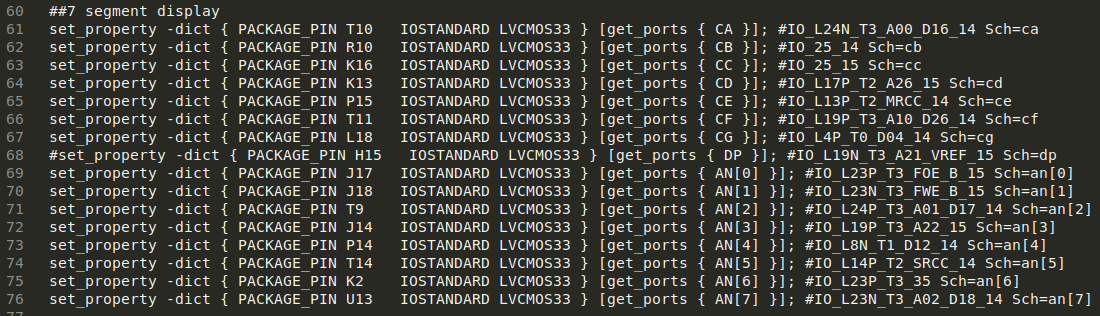


그림 5. 8 자리 7 세그먼트의 연결은 입력을 표시합니다 (파일 *RVfpganexys.xdc*).

우리 시스템의 최상위 모듈 (모듈 **RVfpganexys**, *[RVfpgaPath]/RVfpga/src/RVfpganexys.sv* 파일에 구현 됨)의 50-51 행에서 SoC에 연결된 7 세그먼트 디스플레이 입력 신호를 찾을 수 있습니다. *AN[i]* 및 *CA*… *CG* (그림 6의 왼쪽 부분), 해당 모듈의 끝 (그림 6의 오른쪽 부분)에서 **swervolf\_core** 모듈에 대한 연결을 찾을 수 있습니다 (*CA*… *CG* 신호의 이름은 해당 모듈은 *Digits\_Bits* [6:0])로 표시됩니다.

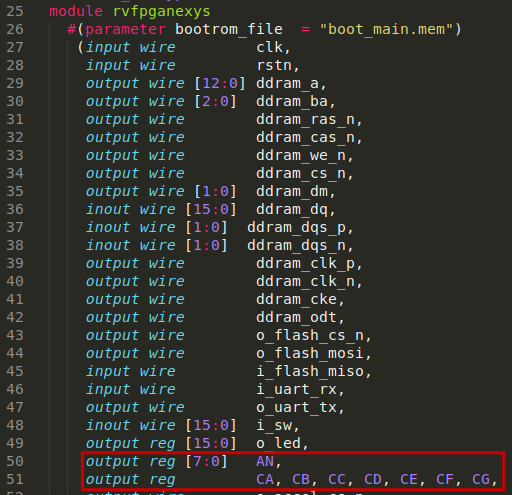
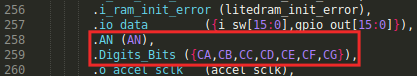
 

그림 6. 8 자리 7 세그먼트 디스플레이를 SoC에 연결 (파일: *rvfpganexys.sv*).

마지막으로 2 개의 신호가 **swervolf\_core** 모듈에서 시스템 컨트롤러 모듈 (**swervolf\_syscon**)로 삽입 됩니다 (그림 7 참조). 여기서 7 세그먼트 디스플레이 컨트롤러가 구현됩니다.

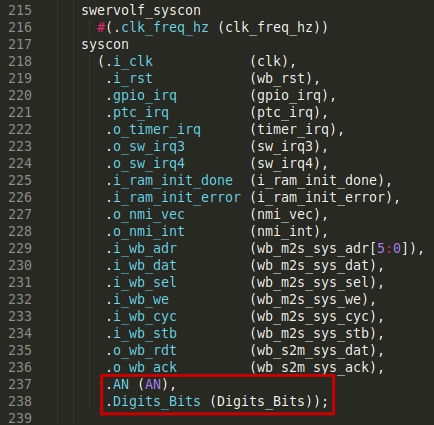


그림 7. 8 자리 7 세그먼트 디스플레이를 시스템 컨트롤러에 연결 (파일: *Swervolf\_core.v*).

**작업:** 제약 조건 파일에서 시스템 컨트롤러 모듈 (*CA-CG*가 *Digits\_Bits* 배열로 병합 됨)으로 이 신호 (*CA-CG* 및 *AN*)를 따르십시오. 다음 파일을 확인하시기 바랍니다.

*[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. **8 자리 7 세그먼트 디스플레이 컨트롤러를 SoC에 통합**

모듈 **swervolf\_syscon** (*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf\_syscon.v*)의 276-288 행에서 7 세그먼트 디스플레이 컨트롤러가 인스턴스화되고 SoC에 통합됩니다 (그림 8 참조).

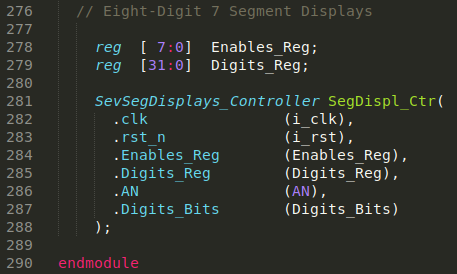


그림 8. 8 자리 7 세그먼트는 컨트롤러 인스턴스화를 표시합니다 (파일: *swervolf\_syscon.v*).

**SevSegdisplays\_Controller** 모듈은 클럭 신호 (*i\_clk, clk*로 이름 변경) 및 리셋 신호 (*i\_rst, rst\_n*으로 이름 변경) 외에도 두 개의 입력 신호 (*Enables\_Reg* 및 *Digits\_Reg*)를 수신합니다. 이 신호는 이미 설명한 두 개의 메모리 매핑 제어 레지스터입니다. 이 모듈은 보드의 7 세그먼트 디스플레이에 연결된 두 개의 신호 *AN* 및 *Digits\_Bits*를 출력합니다. 맨 오른쪽 두 자리에 71을 표시하는 예의 경우 **SevSegdisplays\_Controller**는 신호 *AN* 및 *Digits\_Bits*에 다음 값을 할당합니다.

* 0 ~ 2ms: 신호 *AN[0]*이 low, 숫자 0 (가장 오른쪽 숫자)이 표시 되도록 합니다. 신호 *Digits\_Bits[5]* 및 *Digits\_Bits[4]* (*CB* 및 *CC*에 해당) 또한 숫자 0 (가장 오른쪽 숫자)에 "1"을 표시하도록 low 입니다. 다른 모든 신호는 high 입니다.
* 2 ~ 4ms: 신호 *AN[1]*이 low, 숫자 1을 표시 할 수 있습니다. *Digits\_Bits[6]*, *Digits\_Bits[5]* 및 *Digits\_Bits[4]* (*CA, CB* 및 *CC*에 해당)는 숫자 1에 "7"을 표시하기 위해 high입니다. 다른 모든 신호는 high 입니다.
* 4 ~ 16ms: *AN[2]*… *AN[7]*은 2ms 간격으로 높으므로 값을 표시하지 않습니다. 세그먼트는 나머지 자릿수인 2-7 자릿수에 대해서도 high 입니다.

**SevSegdisplays\_Controller** 모듈은 *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf\_syscon.v* 파일의 295-366 행에서 구현 됩니다. 여기에는 다음과 같은 서브유닛이 포함됩니다.

* 2 개의 멀티플렉서가 2ms마다 *AN* 및 *Digits\_Bits* 신호로 보낼 값을 선택합니다. 멀티플렉서는 **SevSegMux** 모듈 내부에서 구현됩니다.
* 2ms주기를 생성하기 위해 우리는 *[RVfpgaPath]/RVfpga/src/OtherSources/pulp-platform.org\_\_common\_cells\_1.20.0/src*폴더에 포함된 *counter.sv* 및 *delta\_counter.sv* 파일에 제공된 **카운터** 모듈을 사용합니다. 카운터는 0에서 219까지 카운트하도록 구성되며, 대략 2ms마다 변경되는 3 개의 최상위 비트가 위에서 설명한 두 멀티플렉서의 선택 신호로 사용됩니다.
* 디코더는 주어진 4 비트 16 진수 값에 대한 세그먼트 값을 출력하는 모듈 **SevenSegDecoder**에서 구현됩니다.

**작업: SevSegdisplays\_Controller** 모듈을 자세히 분석합니다. 다음 섹션에서 수행되는 시뮬레이션은이 작업에 도움이 될 수 있습니다. 필요한 경우 새 신호로 시뮬레이션을 확장할 수도 있습니다.

1. **8 자리 7 세그먼트 디스플레이 컨트롤러와 SweRV EH1 Core 연결**

LAB 6에 설명된 대로 장치 컨트롤러는 멀티플렉서를 사용하여 SweRV EH1 Core에 연결됩니다 (그림 4 참조). 7:1 멀티플렉서 (그림 9)는 *[RVfpgaPath]/RVfpga/src* 파일의 104-205 행에서*(/SweRVolfSoC/Interconnect/WishboneInterconnect/wb\_intercon.vh*) 인스턴스화되는 *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Interconnect/WishboneInterconnect/wb\_intercon.v* 파일에서 구현 됩니다. 이후 파일은 [*RVfpgaPath]/RVfpga/src/SweRVolfSoC/swervolf\_core.v* 위치에 있는 **swervolf\_core** 모듈의 168 행에 포함되어 있습니다.

멀티플렉서는 주소에 (110~111) 따라 CPU (*wb\_io\_* \* 신호 – 그림 9의 115-126 행)를 Wishbone 버스 (그림 9의 127-138 행)에 연결하여 읽거나 쓸 주변 장치를 선택합니다. 예를 들어 CPU에서 생성된 주소가 0x80001000-0x8000103F 범위에 있으면 시스템 컨트롤러가 선택되므로 *wb\_io\_* \* 신호는 *wb\_sys\_* \* 신호와 연결됩니다.



그림 9. 7-1 CPU와 연결된 주변 장치를 선택하는 멀티플렉서 (파일: *wb\_intercon.v*).

시스템 컨트롤러에 포함된 레지스터는 CPU에서 생성된 주소 (*i\_wb\_adr*) (**swervolf\_syscon** 모듈의 162-228 라인)를 기반으로 Wishbone 버스 (*i\_wb\_dat*)의 데이터 신호에 직접 연결하여 CPU에서 기록됩니다.

**작업:** 시스템 컨트롤러에서 주소가 매핑되는 방식을 이해하기 위해 모듈 **swervolf\_syscon**의 162-228 행을 검사합니다. 레지스터 *Enables\_Reg* 및 *Digits\_Reg*를 참조하는 219 ~ 227 행 (그림 10)에 중점을 둡니다 (앞에서 언급했듯이 이 두 레지스터에 할당된 주소는 각각 0x80001038 및 0x8000103C 임).

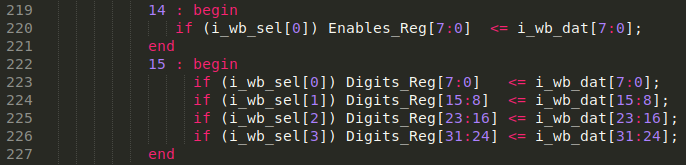


그림 10. 8 자리 7 세그먼트 디스플레이와 Core 간의 연결 (파일 *swervolf\_syscon.v*).

1. **Verilator 시뮬레이션**

이 섹션에서는 프로세서가 이 주변 장치를 구동하는 간단한 예제를 실행할 때 8 자리 7 세그먼트 디스플레이 컨트롤러의 주요 신호를 검사하기 위해 **RVfpgaSim**을 사용합니다. 시뮬레이션에서 우리는 신호 *AN* 및 *Digits\_Bits*를 분석하고 그림 11의 예를 실행하여 맨 오른쪽 두 자리에 71을 씁니다. 이 프로그램은 [RVfpgaPath]/RVfpga/Labs/Lab7/71\_7SegDispl 에서 찾을 수 있습니다. C 버전은 [RVfpgaPath]/RVfpga/Labs/Lab7/71\_7SegDispl\_C-Lang 에서도 찾을 수 있습니다.

|  |
| --- |
| #define SegEn\_ADDR 0x80001038  #define SegDig\_ADDR 0x8000103C  .globl main  main:  li t1, SegEn\_ADDR  li t6, 0xFC  **sb x6, 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 |

그림 11. 71\_7SegDispl.S 예제

그림 12는 71\_7SegDispl.elf 프로그램의 디스 어셈블리 버전을 보여줍니다. 이 프로그램은 PlatformIO에서 컴파일한 후 다음에서 찾을 수 있습니다. [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> |

그림 12. 71\_7SegDispl.S 예제의 디스 어셈블리 버전

# 시뮬레이션을 실행하려면 다음 단계를 따르십시오. (시뮬레이션을 실행하지 않으려면 7 단계로 바로 이동할 수 있습니다.)

1. 이 경우 시뮬레이션의 경우에만 COUNT\_MAX (*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf\_syscon.v* 파일의 295 행 참조)를 20에서 5로 변경하여 클럭 주기를 줄여야 합니다. 그렇지 않으면 결과를 보는 데 너무 많은 시간이 걸립니다. COUNT\_MAX의 값을 수정한 후 다음 명령을 실행하여 RVfpgaSim을 다시 컴파일하십시오 (GSG에 설명되어 있음).

cd [RVfpgaPath]/RVfpga/verilatorSIM

make clean

make

새 파일 *Vrvfpgasim* (RVfpgaSim 시뮬레이션 바이너리)은 *[RVfpgaPath]/RVfpga/verilatorSIM* 디렉토리 내에 생성되어야 합니다.

**WINDOWS**: Windows를 사용하는 경우 Cygwin 터미널에서 이러한 명령을 실행해야 합니다 (자세한 지침은 시작 안내서의 섹션 6 및 부록 C 참조). C:Windows 폴더는 Cygwin (*/cygdrive/c*)에서 찾을 수 있습니다.

**MacOS:** 자세한 지침은 시작 안내서의 부록 D를 참조하십시오.

1. 컴퓨터에서 VSCode/PlatformIO를 엽니다.
2. 상단 표시 줄에서 파일-폴더 열기를 클릭하고 *[RVfpgaPath]/RVfpga/Labs/Lab7* 디렉터리로 이동합니다.
3. *71\_7SegDispl* 디렉토리를 선택하고 (열지 말고 선택 만) 확인을 클릭하십시오. 예제는 PlatformIO에서 열립니다.
4. *platformio.ini* 파일을 열고 위에서 생성한 RVfpgaSim 시뮬레이션 바이너리의 경로가 올바른지 확인합니다. GSG에서 다음과 같이 표시되어야 합니다:

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

1. 왼쪽 메뉴 리본에서 PlatformIO 아이콘을 클릭하여 시뮬레이션을 실행한 다음 Project Tasks → env: swervolf\_nexys → Platform을 확장하고 Generate Trace를 클릭합니다.

*trace.vcd* 파일은 *[RVfpgaPath]/RVfpga/Labs/Lab7/71\_7SegDispl/.pio/build/swervolf\_nexys* 내에 생성되어야 하며 다음 명령을 실행하여 *GTKWave*로 열 수 있습니다.

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

**WINDOWS:** 다운로드한 *gtkwave64* 폴더의 bin 폴더 안에 *gtkwave.exe*라는 응용 프로그램이 포함되어 있습니다. 해당 응용 프로그램을 두 번 클릭하여 GTKWave를 시작합니다. 응용 프로그램의 상단에서 **파일 – 새 탭 열기**를 클릭하고 *[RVfpgaPath]/RVfpga/Labs/Lab7/71\_7SegDispl/.pio/build/swervolf\_nexys*폴더에 생성된 trace.vcd 파일을 엽니다.

1. 시뮬레이션에 다음 신호를 포함합니다 (각 신호를 찾기 위해 참조된 모듈로 이동).

* rvfpgasim – swervolf – syscon – SegDispl\_Ctr
  + 입력 신호: ***Enables\_Reg*** and ***Digits\_Reg***.
  + 출력 신호: ***AN*** and ***Digits\_Bits***.

1. 그림 13에 표시된 시뮬레이션을 분석합니다. 처음에 8 개의 7 세그먼트 디스플레이에 표시된 값은 모두 0입니다 (초기에는 모든 숫자가 *Enables\_Reg*=0으로 활성화 됨). 그런 다음 *Enables\_Reg* (그림 12의 sb 명령어)에 *0xFC*를 작성하여 맨 왼쪽 6 자리 숫자를 비활성화하고 *Digits\_Reg* (그림 12의 sw 명령어)에 *0x71*을 작성하여 맨 오른쪽 2 자리 숫자에 *71*을 작성합니다. 출력 신호에 미치는 영향은 다음과 같습니다 (그림 13 참조).
   * 첫 번째 기간: *AN=0xFE* 및 *Digits\_Bits=0x4F*, 따라서 맨 오른쪽 숫자인 숫자 0에 *1*이 표시됩니다.
   * 두 번째 기간: *AN=0xFD* 및 *Digits\_Bits=0x0F*, 따라서 다음 숫자인 숫자 1에 *7*이 표시됩니다.
   * 다음 6 개 기간: *AN=0xFF* 및 *Digits\_Bits =0x01*, 따라서 가장 왼쪽에 있는 6 자리 숫자를 끕니다.
   * 이 과정이 반복됩니다.

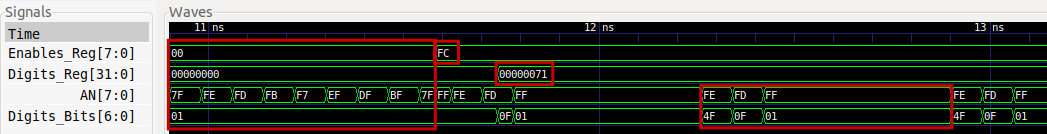


그림 13. 8 자리 7 세그먼트 디스플레이의 가장 오른쪽 두 자리에 값 71 쓰기

1. 계속하기 전에 COUNT\_MAX의 값을 원래 값 (COUNT\_MAX = 20)으로 복원하는 것을 잊지 마십시오.

# 고급 실습

# 연습 3. 7 세그먼트 디스플레이에 ON/OFF LED 조합을 표시할 수 있도록 이번 LAB에서 설명한 컨트롤러를 수정합니다.

* 지금 활성화 레지스터가 필요하지 않습니다. 대신 8 개의 7 비트 레지스터가 필요합니다. 8 자리 7 세그먼트 디스플레이 각각의 세그먼트를 Segments\_Digit0 ~ Segments\_Digit7라 부릅니다. 각 레지스터에서 각 비트는 해당 세그먼트가 ON (0)인지 OFF (1)인지를 나타냅니다. 예를 들어, 첫 번째 레지스터 (Segments\_Digit0)의 모든 비트가 0이면 맨 오른쪽 숫자의 모든 세그먼트가 ON이되고 첫 번째 레지스터의 모든 비트가 1이면 맨 오른쪽 숫자의 모든 세그먼트가 OFF 됩니다.
* 이 두 개의 새 레지스터를 이전에 사용한 것과 동일한 주소에 매핑할 수 있습니다 (먼저 이전 레지스터 *Enables\_Reg* 및 *Digits\_Reg* 두 개를 제거).
  + Segments\_Digit0 🡨🡪 Address 0x80001038
  + Segments\_Digit1 🡨🡪 Address 0x80001039
  + …
  + Segments\_Digit7 🡨🡪 Address 0x8000103F
* 프로그램에서 제공하는 정보가 이미 디코딩 되었으므로 더 이상 4-7 디코더 (모듈 **SevenSegDecoder**)가 필요하지 않습니다.

# 연습 4. 새 컨트롤러를 사용하여 8 자리 7 세그먼트 디스플레이에 "I SAY HI"를 인쇄합니다. 평소와 같이 프로그램의 RISC-V 어셈블리와 C 버전을 모두 구현합니다.

1. The information in this section is described in: <https://reference.digilentinc.com/reference/programmable-logic/nexys-a7/reference-manual> [↑](#footnote-ref-1)