



THE IMAGINATION UNIVERSITY PROGRAMME

RVfpga Lab 7

Mostrador de 7-Segmentos

1. INTRODUÇÃO

Este laboratório descreve como o Sistema RVfpga foi alargado para funcionar com mostradores de 7 segmentos e mostra como modificar o controlador de mostrador de 7 segmentos. A placa Nexys A7 FPGA tem oito mostradores de 7 segmentos. Descrevemos primeiro como funcionam (Secção 2) e depois analisamos a especificação de alto nível do controlador do mostrador de 8 dígitos de 7 segmentos incluído no Sistema RVfpga e fornecemos alguns exercícios elementares (Secções 3 e 4). Finalmente, analisamos a implementação de baixo nível deste controlador, realizamos uma simulação de Verilator e fornecemos Exercícios adicionais onde modificará e experimentará a implementação do controlador (Secções 5 e 6).

2. MOSTRADORES DE 7 SEGMENTOS NA PLACA NEXYS A7

A placa Nexys A7 contém dois blocos de 4 dígitos de mostradores a LED de 7-segmentos com ânodo comum¹, configurado para se comportar como um único mostrador de 8 dígitos de 7 segmentos (ver Figura 1). Cada um dos oito dígitos é composto por sete segmentos dispostos no padrão "8" (ver Figura 2), com um LED para cada segmento. Cada um destes segmentos pode ser ligado ou desligado, pelo que qualquer um dos 128 padrões pode ser exibido num dígito, iluminando certos segmentos LED e deixando os outros apagados; especificamente, entre estes 128 padrões, os dígitos decimais podem ser exibidos como mostrado na Figura 2.

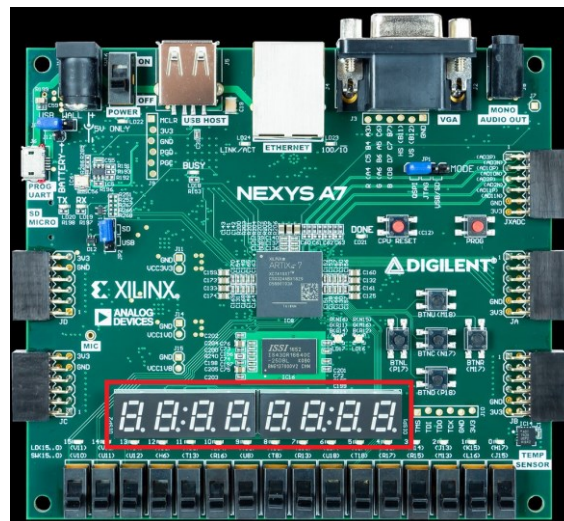


Figura 1. Mostrador de 8 dígitos de 7 segmentos na Nexys A7

¹ A informação nesta secção é descrita em:

<https://reference.digilentinc.com/reference/programmable-logic/nexys-a7/reference-manual>

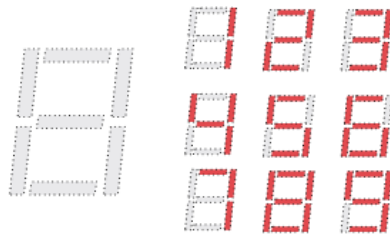


Figura 2. Padrões correspondentes aos dígitos decimais

Os segmentos LED de um único dígito estão etiquetados A-G, como mostra à direita da Figura 3. Os ânodos dos sete LEDs de um único dígito estão ligados num único nó de circuito "ânodo comum", mas os cátodos de LEDs permanecem separados (ver Figura 3). Os oito sinais dos ânodos comuns, um para cada dígito (*AN0-AN7*), funcionam como uma "habilitação de dígitos". Os cátodos do mesmo segmento em todos os oito dígitos estão ligados em sete sinais, *CA-CG* (ver Figura 3). (Note-se que existe um oitavo sinal para o ponto decimal, *DP*, mas não o utilizaremos neste laboratório.) Por exemplo, o cátodo do segmento D dos oito dígitos é agrupado num único nó de circuito chamado CD. Este esquema de ligação do sinal cria um mostrador multiplexado, onde os sinais dos cátodos são comuns a todos os dígitos, mas só podem iluminar os segmentos do dígito cujo sinal do ânodo correspondente é ativo. Todos estes sinais são ativados com zero ou nível lógico baixo; assim, para iluminar um segmento, por exemplo, o segmento D no dígito 2, tanto o ânodo *AN2* como o cátodo CD devem ser ativos a nível lógico baixo.

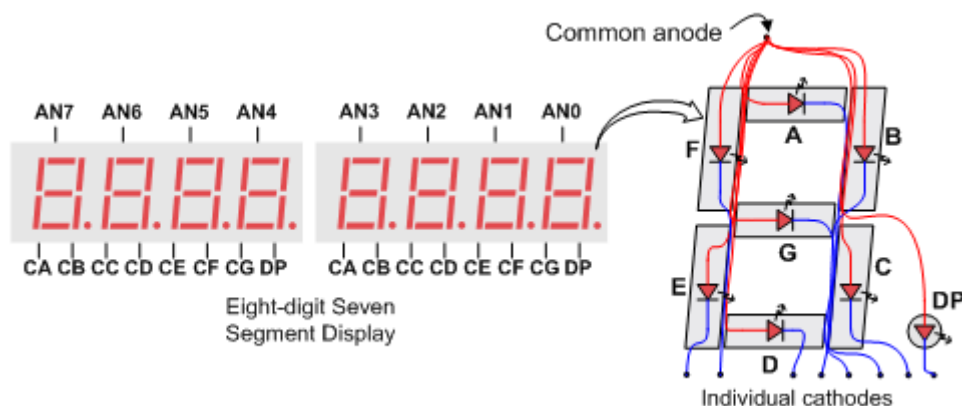


Figura 3. Ligação do mostrador de 8 dígitos de 7 segmentos no Nexys A7

Um circuito controlador de varrimento do mostrador pode ser usado para mostrar um número com 8 dígitos no mostrador de 8 dígitos e 7 segmentos. Este circuito ativa os cátodos com o padrão de cada dígito numa sucessão repetida e contínua a uma taxa de actualização mais rápida do que o olho humano consegue detectar; ao mesmo tempo, o circuito ativa um ânodo de cada vez. Assim, cada dígito é iluminado apenas um oitavo do tempo, mas, como o olho não consegue perceber o escurecimento de um dígito antes de ser iluminado novamente, os dígitos parecem estar continuamente iluminados.

Para que cada um dos 8 dígitos apareça brilhante e continuamente iluminado, todos os 8 dígitos devem ser ativados uma vez a cada 1-16 ms, e cada dígito deverá ser iluminado durante 1/8 do ciclo de actualização (por exemplo, para um ciclo de actualização de 16ms, cada dígito é iluminado durante 2ms). Como explicado acima, o controlador deve acionar os cátodos de um dígito com o padrão correto, enquanto o sinal do ânodo correspondente

também é acionado no nível lógico baixo. Contudo, uma vez que o Nexys A7 utiliza transístores NPN para conduzir corrente suficiente para o ponto de ânodo comum, os ânodos habilitados são invertidos. Portanto, tanto o sinal AN0...7 como o sinal CA...G/DP são ativados para nível lógico baixo quando ativos.

Para ilustrar o processo, suponha que quer mostrar 71 nos dois dígitos mais à direita. O circuito controlador ativaria AN0, CB, e CC baixo durante os primeiros 2ms, mostrando assim um 1 no dígito mais à direita. Depois, durante os 2ms seguintes, o circuito ativaria AN1, CA, CB, e CC baixo, mostrando assim um 7 no dígito mais significativo seguinte. Se o processo for repetido indefinidamente, o olho humano verá o número 71 nos dois dígitos mais à direita.

3. ESPECIFICAÇÃO DE ALTO NÍVEL DO CONTROLADOR DO MOSTRADOR DE 8 DÍGITOS DE 7 SEGMENTOS

Nesta secção, descrevemos e analisamos primeiro a especificação de alto nível do controlador do mostrador de 8 dígitos de 7 segmentos utilizado no Sistema RVfpga, e depois fornecemos exercícios para a sua utilização.

A. Especificação de Alto-nível

O controlador do mostrador de 8 dígitos de 7 segmentos utilizado neste curso foi concebido à medida para o Sistema RVfpga. Inclui dois registos, chamados *Enables_Reg* e *Digits_Reg*, que são mapeados para os endereços 0x80001038 e 0x8000103C respectivamente (note que estes endereços são endereços livres dentro da gama de endereços reservados para o Controlador do Sistema, que pode visualizar em <https://github.com/chipsalliance/Cores-SweRVolf>).

TAREFA: Localizar a declaração dos registos *Enables_Reg* e *Digits_Reg*, bem como o local onde lhes é atribuído um valor. O mostrador de 8 dígitos de 7 segmentos é implementado em ficheiro:
`[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf_syscon.v`.

Enables_Reg é um registo de 8 bits onde cada bit determina se o dígito correspondente está ligado (ON / 0) ou desligado (OFF / 1). *Digits_Reg* é um registo de 32 bits onde cada grupo de 4 bits representa o valor hexadecimal para mostrar no dígito correspondente. Por exemplo, para mostrar 71 nos dois dígitos mais à direita, o programador tem de atribuir os seguintes valores aos registos:

- *Enables_Reg* = 0xFC (dois dígitos mais à direita ativados)
- *Digits_Reg* = 0x00000071 (valor = 71)

4. EXERCÍCIOS ELEMENTARES

Exercício 1. Escreva um programa Assembly RISC-V e um programa em C que mostram o valor dos interruptores nos quatro dígitos mais à direita dos mostradores de 7 segmentos.

Exercício 2. Escrever um programa Assembly RISC-V e um programa em C que mostram a sequência "0-1-2-3-4-5-6-7-8" movendo-se da direita para a esquerda nos mostradores de 8 dígitos de 7 segmentos. Ou seja, 0 deve aparecer primeiro no dígito mais à direita. Depois deve mover-se para a esquerda e 1 deve aparecer no dígito mais à direita, e assim sucessivamente.

5. CONTROLADOR DE MOSTRADOR DE 8 DÍGITOS DE 7-SEGMENTOS: IMPLEMENTAÇÃO DE BAIXO NÍVEL, SIMULAÇÃO

Até este ponto, mostrámos como utilizar apenas os mostradores de 8 dígitos de 7 segmentos. Nesta secção, descrevemos a sua implementação de baixo nível e analisamos o RVfpgaSim em simulação ao executar um simples exemplo em Assembly. Finalmente, fornecemos Exercícios para modificar o controlador de visualização de 8 dígitos de 7 segmentos.

A. Implementação de baixo nível do controlador do mostrador de 8 dígitos de 7 segmentos

À semelhança dos anteriores laboratórios de E/S de uso geral (GPIO), dividimos a análise do controlador do mostrador de 8 dígitos de 7 segmentos em três fases:

1. Ligação entre o SoC e o dispositivo de E/S na placa (região sombreada à esquerda na Figura 4);
2. Integração do novo controlador, que está incluído dentro do controlador do sistema SweRVolfX contido no SoC (região sombreada do meio na Figura 4);
3. Ligação entre o novo controlador e o SweRV EH1 Core (região sombreada à direita na Figura 4).

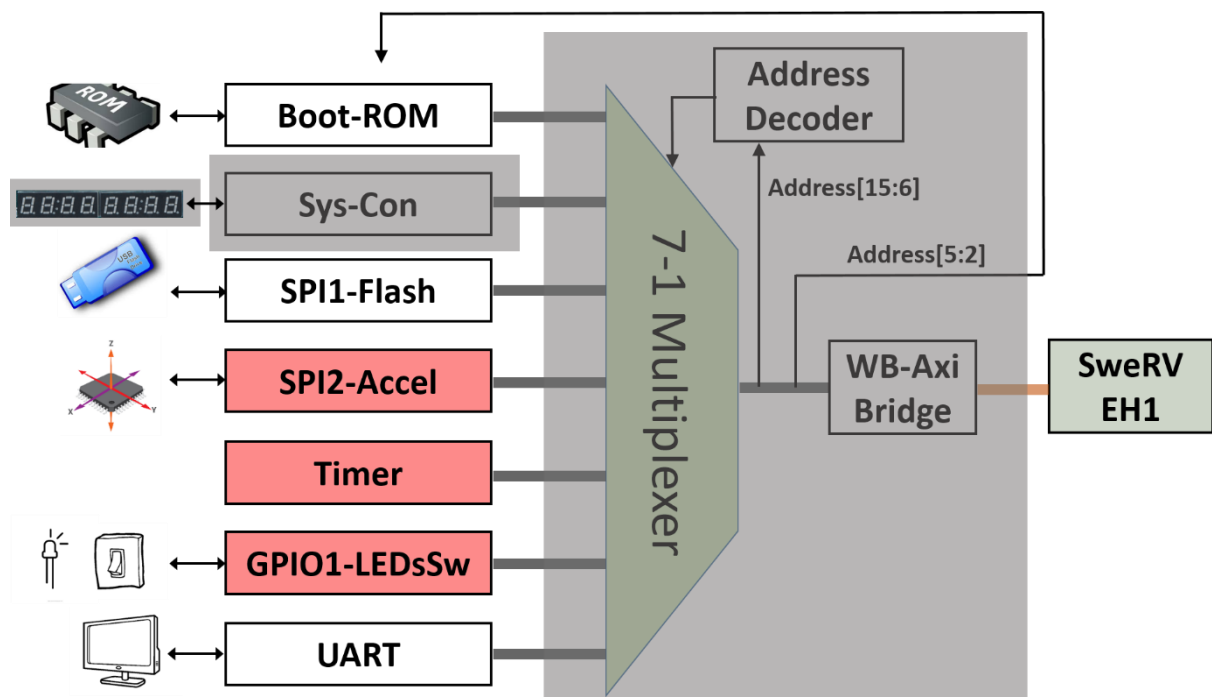


Figura 4. Análise do controlador de 8 dígitos de 7 segmentos em 3 fases

1. Ligação dos LEDs/Switches ao SoC

Finalmente, os dois sinais são inseridos a partir do módulo **swervolf_core** no modulo System Controller (**swervolf_syscon**) (ver Figura 7), onde o controlador de mostrador de 8 dígitos de 7 segmentos estiver implementado.

```

215     swervolf_syscon
216     #(.clk_freq_hz (clk_freq_hz))
217     syscon
218     (.i_clk          (clk),
219      .i_rst          (wb_rst),
220      .gpio_irq       (gpio_irq),
221      .ptc_irq        (ptc_irq),
222      .o_timer_irq    (timer_irq),
223      .o_sw_irq3       (sw_irq3),
224      .o_sw_irq4       (sw_irq4),
225      .i_ram_init_done (i_ram_init_done),
226      .i_ram_init_error (i_ram_init_error),
227      .o_nmi_vec       (nmi_vec),
228      .o_nmi_int       (nmi_int),
229      .i_wb_addr       (wb_m2s_sys_addr[5:0]),
230      .i_wb_dat        (wb_m2s_sys_dat),
231      .i_wb_sel        (wb_m2s_sys_sel),
232      .i_wb_we         (wb_m2s_sys_we),
233      .i_wb_cyc        (wb_m2s_sys_cyc),
234      .i_wb_stb        (wb_m2s_sys_stb),
235      .o_wb_rdt        (wb_s2m_sys_dat),
236      .o_wb_ack        (wb_s2m_sys_ack),
237      .AN (AN),
238      .Digits_Bits (Digits_Bits));
239

```

Figura 7. Ligação das entradas do mostrador de 8 dígitos de 7 segmentos ao System Controller (ficheiro: swervolf_core.v).

TAREFA: Siga estes sinais (CA-CG e AN) do ficheiro de *constraints* para o modulo do System Controller (onde CA-CG são agrupados no barramento *Digits_Bits*). Terá de inspecionar os seguintes ficheiros:

```

[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

```

2. Integração do mostrador de 8 dígitos de 7 segmentos com o SoC

Nas linhas 276-288 do módulo **swervolf_syscon** (*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf_syscon.v*) o controlador de mostrador de 8 dígitos de 7 segmentos é instanciado e integrado no SoC (ver Figura 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

```

Figura 8. Instanciação do controlador do mostrador de 8 dígitos de 7 segmentos (ficheiro: *swervolf_syscon.v*).

O módulo **SevSegdisplays_Controller** recebe, além do sinal do relógio (*i_clk*, renomeado como *clk*) e do sinal de reset (*i_rst*, renomeado como *rst_n*), dois sinais de entrada (*Enables_Reg* e *Digits_Reg*), que são os dois registos de controlo mapeados em memória já descritos. Este módulo gera dois sinais, *AN* e *Digits_Bits*, que estão ligados aos mostradores de 7 segmentos na placa. Para o exemplo mostrar 71 nos dois dígitos mais à direita, o **SevSegdisplays_Controller** atribuiria os seguintes valores aos sinais *AN* e *Digits_Bits*:

- De 0 a 2ms: O sinal *AN[0]* é posto a nível baixo ("0") para ativar o dígito 0 (o dígito mais à direita). Os sinais *Digits_Bits[5]* e *Digits_Bits[4]* (que correspondem a *CB* e *CC*) são também 0 para ativar "1" no dígito 0 (o dígito mais à direita). Todos os outros sinais são colocados a nível alto ("1").
- De 2 a 4ms: O sinal *AN[1]* é colocado a nível baixo para ativar o dígito 1. *Digits_Bits[6]*, *Digits_Bits[5]* e *Digits_Bits[4]* (que correspondem a *CA*, *CB*, e *CC*) são colocados a nível lógico alto para exibir "7" no dígito 1. Todos os outros sinais estão no nível lógico alto.
- De 4 a 16ms: *AN[2]...AN[7]* ficam a nível lógico alto durante intervalos de 2 ms para que não apresentem valores. Os segmentos também são colocados a nível lógico alto para os restantes dígitos (2-7).

O módulo **SevSegdisplays_Controller** é implementado nas linhas 295-366 de ficheiro *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf_syscon.v*. Contém as seguintes subunidades:

- Dois multiplexers selecionam o valor a enviar para os sinais *AN* e *Digits_Bits* a cada 2ms. O multiplexer é implementado dentro do módulo **SevSegMux**.
- Para criar o período de 2ms, usamos um módulo **counter** fornecido nos ficheiros *counter.sv* e *delta_counter.sv*, ambos incluídos na pasta *[RVfpgaPath]/RVfpga/src/OtherSources/pulp-platform.org__common_cells_1.20.0/src*. O contador é configurado para contar a de 0 até 2^{19} , e os 3 bits mais significativos, que mudam aproximadamente a cada 2ms, são usados como os sinais de seleção para os dois multiplexers descritos acima.
- Um descodificador é implementado em módulo **SevenSegDecoder**, que produz os valores do segmento para um valor hexadecimal de 4 bits.

TAREFAS: Analisar o módulo **SevSegdisplays_Controller** em detalhe. A simulação realizada na secção seguinte pode ajudá-lo nesta tarefa. Também pode estender a simulação com novos sinais, se necessário.

3. Ligação entre o controlador do mostrador de 8 dígitos de 7 segmentos e o Core SweRV EH1

Como descrito no Lab 6, os controladores do dispositivo estão ligados ao Core SweRV EH1 utilizando um multiplexer (ver Figura 4). Lembre-se que o multiplexer 7:1 (Figura 9) está implementado no ficheiro *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Interconnect/WishboneInterconnect/wb_intercon.v*, que é instanciado nas linhas 104-205 do ficheiro *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Interconnect/WishboneInterconnect/wb_intercon.vh*. Este último ficheiro está incluído na linha 168 do módulo **swervolf_core** localizado aqui: *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/swervolf_core.v*.

O multiplexer seleciona qual periférico para ler ou escrever, ligando o CPU (sinais *wb_io_** – linhas 115-126 da Figura 9) com o barramento Wishbone de um periférico (linhas 127-138 da Figura 9), dependendo do endereço (linhas 110-111). Por exemplo, se o endereço gerado pelo CPU estiver na gama 0x80001000-0x8000103F, o System Controller é selecionado, e assim os sinais *wb_io_** serão ligados com os sinais *wb_sys_**.

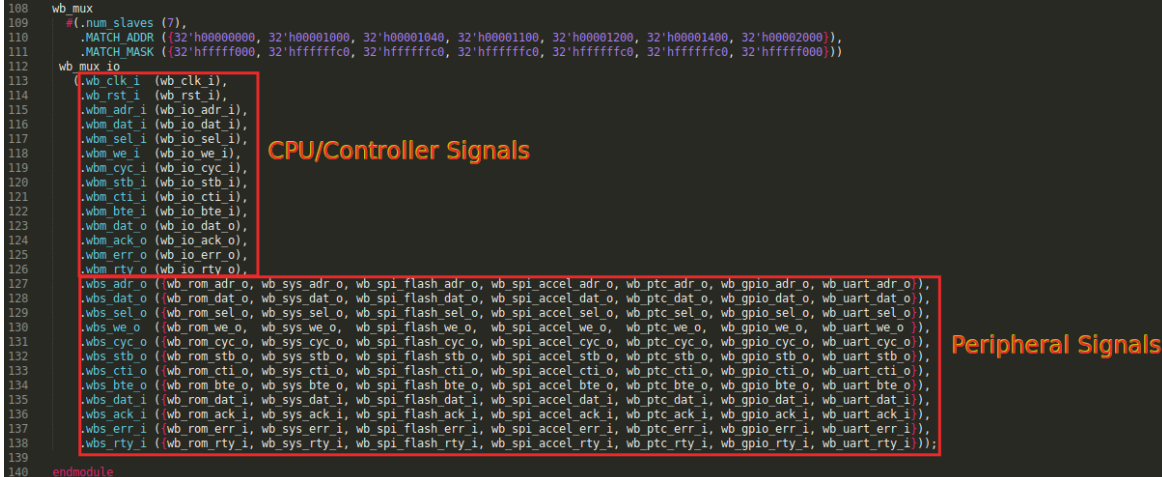


Figura 9. Multiplexer 7-1 que seleciona o periférico ligado ao CPU (ficheiro: *wb_intercon.v*).

Os registos incluídos no System Controller são escritos a partir do CPU ligando-os diretamente ao sinal de dados do barramento Wishbone (*i_wb_dat*), com base no endereço (*i_wb_adr*) gerado pelo CPU (linhas 162-228 do módulo **swervolf_syscon**).

TAREFA: Inspeccione as linhas 162-228 do módulo **swervolf_syscon** para entender como os endereços são mapeados no System Controller. Concentre-se nas linhas 219 a 227 (Figura 10), que se referem aos registos *Enables_Reg* e *Digits_Reg* (como mencionamos anteriormente, os endereços atribuídos a estes dois registos são 0x80001038 e 0x8000103C respectivamente).

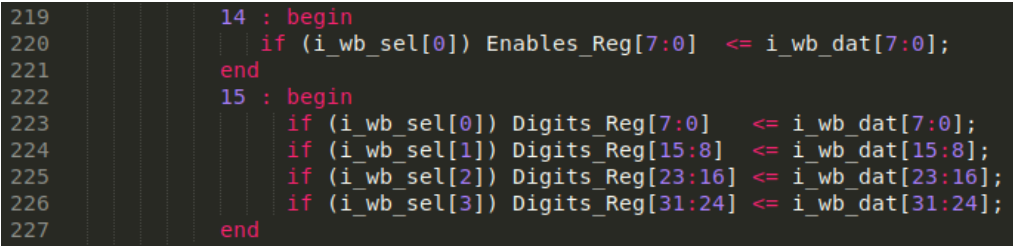


Figura 10. Ligação entre o mostrador de 8 dígitos de 7 segmentos e o Core (ficheiro *swervolf_syscon.v*).

B. Simulação em Verilator

Nesta secção, usamos o **RVfpgaSim** para inspecionar os principais sinais do controlador de 8 dígitos de 7 segmentos, quando o processador executa um exemplo simples que controla este periférico. Na simulação, analisamos os sinais *AN* e *Digits_Bits* enquanto executamos

o exemplo da Figura 11, que escreve 71 nos dois dígitos mais à direita. Pode encontrar este programa em: [RVfpgaPath]/RVfpga/Labs/Lab7/71_7SegDispl (pode também encontrar a versão C em: [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)                # Ativa 7SegDisplays

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

next: beq zero, zero, next

.end
```

Figura 11. Exemplo 71_7SegDispl.S

A Figura 12 mostra a versão “desassemblada” do programa 71_7SegDispl.elf, que, após a compilação no PlatformIO, pode-se encontrar em:

[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>
```

Figura 12. Versão “desassemblada” do exemplo 71_7SegDispl.S

Siga os próximos passos para executar a simulação. (Se preferir não executar a simulação, pode ir diretamente para o passo 7.)

1. Neste caso, e apenas para a simulação, deve reduzir o período do relógio mudando COUNT_MAX (ver linha 295 do ficheiro [RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf_syscon.v) de 20 para 5; caso contrário, levaria muito tempo para observar os resultados. Modifique o valor de COUNT_MAX e, em seguida, recompile o RVfpgaSim executando os seguintes comandos (isto está explicado no GSG):


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

Um novo ficheiro *Vrvfpgasim* (o binário de simulação RVfpgaSim), deve ser gerado dentro da pasta *[RVfpgaPath]/RVfpga/verilatorSIM*.

WINDOWS: no caso de estar a utilizar o Windows, tem de executar estes comandos dentro do terminal Cygwin (consulte a Secção 6 e o Apêndice C do Guia de Iniciação para as instruções detalhadas). A pasta C: do Windows encontra-se dentro do Cygwin em: */cygdrive/c*.

MacOS: Consulte o Apêndice D do Guia de Iniciação para as instruções detalhadas.

2. Abra o VSCode/PlatformIO on o seu computador.
3. Na barra superior, clique em *File - Open Folder...*, e navegue até ao diretório *[RVfpgaPath]/RVfpga/Labs/Lab7*
4. Selecione a pasta *71_7SegDispl* (não o abra, mas apenas o selecione) e clique em OK. O exemplo abrirá no PlatformIO.
5. Abra o ficheiro *platformio.ini* e verificar se o caminho para o binário de simulação RVfpgaSim está correcto. Lembrando o Guia de Introdução, deverá ser parecido com:


```
board_debug.verilator.binary =
  [RVfpgaPath]/RVfpga/verilatorSIM/Vrvfpgasim
```
6. Executar a simulação clicando no ícone PlatformIO na fita do menu esquerdo , em seguida, expandir Project Tasks → env:swervolf_nexys → Platform e clicar em Generate Trace.

O ficheiro *trace.vcd* deve ter sido gerado no interior *[RVfpgaPath]/RVfpga/Labs/Lab7/71_7SegDispl/.pio/build/swervolf_nexys*, e pode abri-lo com *GTKWave* executando o seguinte comando:

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

WINDOWS: a pasta *gtkwave64* descarregada, inclui uma aplicação chamada *gtkwave.exe* dentro da pasta *bin*. Lance o GTKWave clicando duas vezes nessa aplicação. Na parte superior da aplicação, clique em **File – Open New Tab**, e abrir o ficheiro *trace.vcd* gerado nesta pasta *[RVfpgaPath]/RVfpga/Labs/Lab7/71_7SegDispl/.pio/build/swervolf_nexys*.

7. Inclua os seguintes sinais na simulação (vá para os módulos referidos para localizar cada um dos sinais):
 - rvfpgasim – swervolf – syscon – SegDispl_Ctr
 - ✓ Sinais de entrada: **Enables_Reg** e **Digits_Reg**.
 - ✓ Sinais de saída: **AN** e **Digits_Bits**.
8. Analise a simulação mostrada na Figura 13. Inicialmente, os valores mostrados nos oito mostradores de 7 segmentos estão todos a 0 (inicialmente todos os dígitos são ativados porque *Enables_Reg*=0). Em seguida, desativamos os seis dígitos mais à esquerda, escrevendo *0xFC* para o registo *Enables_Reg* (instrução *sb* na Figura 12) e escrevendo *71* nos dois dígitos mais à direita, escrevendo *0x71* em *Digits_Reg* (instrução *sw* na Figura 12). O efeito nos sinais de saída é o seguinte (como mostrado na Figura 13):

- No primeiro período: $AN=0xFE$ e $Digits_Bits=0x4F$, exibindo assim 1 no dígito mais à direita, dígito 0.
- No segundo período: $AN=0xFD$ e $Digits_Bits=0x0F$, exibindo assim 7 no próximo dígito, dígito 1.
- Nos seis períodos seguintes: $AN=0xFF$ e $Digits_Bits=0x01$, desligando assim os seis dígitos mais à esquerda.
- Este processo, em seguida, repete-se.

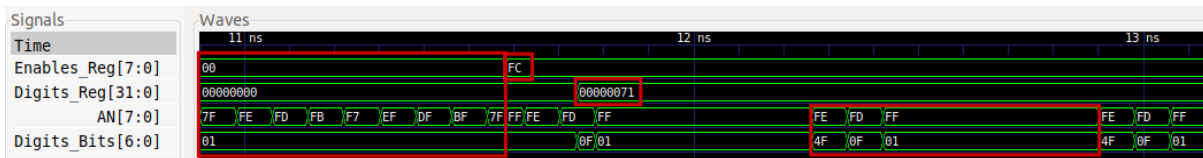


Figura 13. Escreva o valor 71 nos dois dígitos mais à direita do mostrador de 8 dígitos de 7 segmentos

9. Antes de continuar, não se esqueça de restaurar o valor de `COUNT_MAX` ao seu valor original (`COUNT_MAX=20`).

6. EXERCÍCIOS AVANÇADOS

Exercício 3. Modificar o controlador descrito neste laboratório para que o mostrador de 8 dígitos de 7 segmentos possa mostrar qualquer combinação de LEDs ON/OFF.

- Não é necessário um registo de habilitação agora. Em vez disso, precisa de oito registos de 7 bits. Chame-os: `Segments_Digit0` – `Segments_Digit7`, um para cada um dos oito mostradores de 7 segmentos. Em cada um destes registos, cada bit indica se o segmento correspondente está ON (0) ou OFF (1). Por exemplo, se todos os bits do primeiro registo (`Segments_Digit0`) forem 0, todos os segmentos do dígito mais à direita serão ON, enquanto todos os bits do primeiro registo forem 1, todos os segmentos do dígito mais à direita serão OFF.
- - É possível mapear estes dois novos registos para os mesmos endereços que utilizámos anteriormente (primeiro remover os dois registos anteriores `Enables_Reg` e `Digits_Reg`):
 - `Segments_Digit0` ↔ Endereço 0x80001038
 - `Segments_Digit1` ↔ Endereço 0x80001039
 - ...
 - `Segments_Digit7` ↔ Endereço 0x8000103F
- Note que já não precisa do descodificador 4-7 (módulo **SevenSegDecoder**), como a informação fornecida pelo programa já está descodificada.

Exercício 4. Utilize o novo controlador para imprimir o seguinte no mostrador de 8 dígitos de 7 segmentos: "I SAY HI". Como habitualmente, implementar tanto as versões Assembly RISC-V como o C deste programa.