



PROGRAMA UNIVERSITARIO DE IMAGINATION

Práctica 7 RVfpga

Displays de 7 segmentos

1. INTRODUCCIÓN

Esta práctica describe cómo se extendió el Sistema RVfpga para trabajar con los displays de 7 segmentos y cómo modificar el controlador de los mismos. La placa FPGA Nexys A7 tiene ocho displays de 7 segmentos. En primer lugar se describe cómo funcionan (Sección 2) y luego se analiza la especificación de alto nivel del controlador del display de 7 segmentos de 8 dígitos que se incluye en el Sistema RVfpga y se proponen algunos ejercicios básicos (Secciones 3 y 4). Finalmente, se analiza la implementación de bajo nivel de este controlador, se realiza una simulación en Verilator y se proponen ejercicios adicionales en los que se modificará y experimentará con la implementación del controlador (Secciones 5 y 6).

2. DISPLAYS DE 7 SEGMENTOS DE LA PLACA NEXYS A7

La placa Nexys A7 contiene dos displays de 7 segmentos LED, de 4 dígitos y ánodo común¹, configurados como un único display de 7 segmentos de 8 dígitos (Figura 1). Cada uno de los 8 dígitos se compone de 7 segmentos dispuestos en un patrón de "figura de 8" (Figura 2), con un LED para cada segmento. Cada uno de estos segmentos puede estar activo o no, de modo que cualquiera de los 128 patrones se puede mostrar en un dígito mediante el encendido de ciertos segmentos LED y el apagado de los restantes; en particular, entre estos 128 patrones, se pueden mostrar los dígitos decimales (Figura 2).

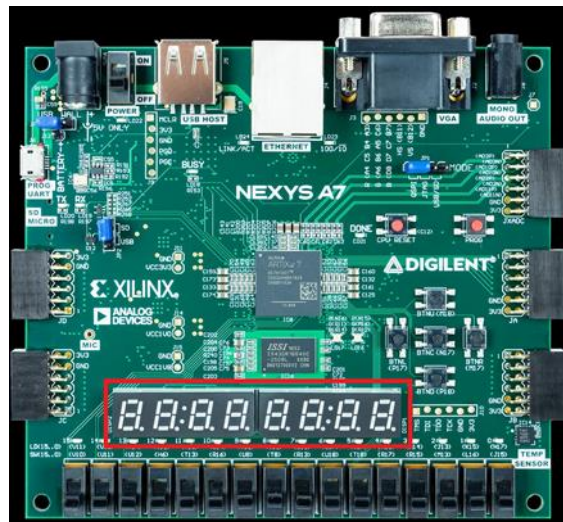
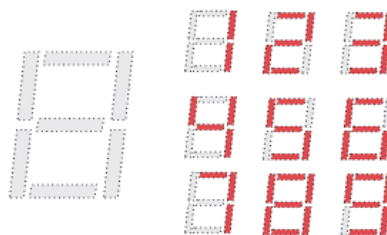


Figura 1. Displays de 7 segmentos de 8 dígitos en la placa Nexys A7



¹ La información de esta sección se describe en:

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

Figura 2. Patrones correspondientes a los dígitos decimales

Los segmentos LED de un solo dígito se etiquetan como A-G, según se muestra en la parte derecha de la Figura 3. Los ánodos de los siete LEDs de un dígito están unidos en un único nodo, de ahí que se denomine "ánodo común", pero los cátodos de los LEDs permanecen separados (ver Figura 3). Las ocho señales del ánodo común, una para cada dígito (AN0-AN7), actúan como "habilitadores de dígitos". Los cátodos del mismo segmento en los ocho dígitos se conectan en siete señales, CA-CG (ver Figura 3). (Obsérvese que existe una octava señal para el punto decimal, DP, pero no se usará en esta práctica). Por ejemplo, el cátodo del segmento D de los ocho dígitos se agrupan en un único nodo del circuito llamado CD. Este esquema de conexión de señales crea un display multiplexado, en el que las señales del cátodo son comunes a todos los dígitos, pero sólo pueden iluminar los segmentos del dígito cuya señal de ánodo correspondiente se activa. Todas estas señales se activan a bajas; así, para iluminar un segmento, por ejemplo el segmento D del dígito 2, tanto el ánodo AN2 como el cátodo CD deben estar a bajas (valor 0).

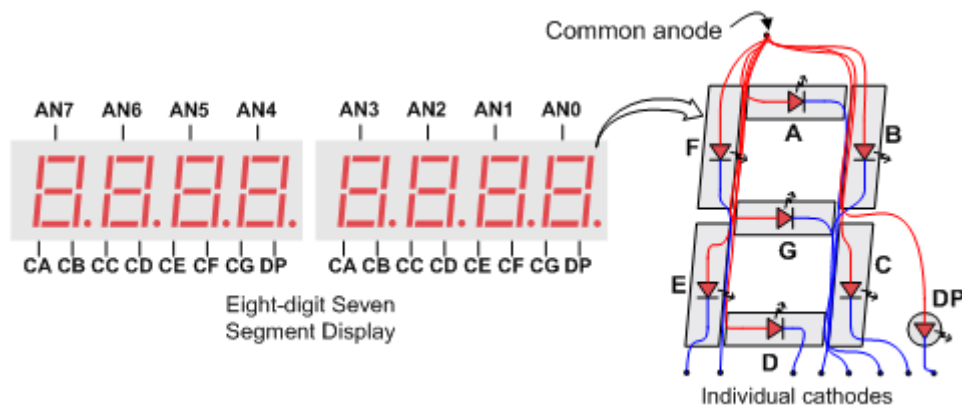


Figura 3. Conexión del display de 7 segmentos de 8 dígitos en la placa Nexys A7

Se puede usar un circuito controlador de display por muestreo para mostrar un número de 8 dígitos en los displays de 7 segmentos de 8 dígitos. Este circuito configura los niveles de tensión de los cátodos con el patrón de cada dígito en una sucesión continua y repetitiva a una velocidad de actualización que es más rápida de lo que el ojo humano puede detectar; al mismo tiempo el circuito configura los ánodos uno a uno. Así, cada dígito se ilumina sólo una octava parte del tiempo, pero, como el ojo no puede percibir el oscurecimiento de un dígito antes de que se ilumine de nuevo, el dígito parece estar continuamente iluminado.

Para que cada uno de los 8 dígitos aparezca encendido y continuamente iluminado, los ocho dígitos deben ser configurados una vez cada 1 a 16 ms, y cada dígito se iluminará durante 1/8 del ciclo de refresco (por ejemplo, para un ciclo de refresco de 16ms, cada dígito se ilumina durante 2ms). Como se ha explicado anteriormente, el controlador debe poner un nivel bajo de tensión en los cátodos de un dígito con el patrón correcto, mientras que en la señal del ánodo correspondiente también se pone un nivel bajo. Sin embargo, como la placa Nexys A7 utiliza transistores NPN para suministrar suficiente corriente al punto del ánodo común, las habilitaciones del ánodo se invierten. Por lo tanto, tanto las señales AN0...7 y CA...G/DP se llevan a bajas cuando están activas.

Para ilustrar el proceso, suponga que quiere mostrar 71 en los dos dígitos de la derecha. El circuito controlador debería poner en bajo AN0, CB y CC durante los primeros 2ms, mostrando así un 1 en el dígito de más a la derecha. Luego, durante los siguientes 2ms, el circuito pondría en bajo AN1, CA, CB y CC, mostrando así un 7 en el siguiente dígito más

significativo. Si el proceso se repite indefinidamente, el ojo humano verá el número 71 en los dos dígitos de la derecha.

3. ESPECIFICACIÓN DE ALTO NIVEL DEL CONTROLADOR DEL DISPLAY DE 7 SEGMENTOS DE 8 DÍGITOS

En esta sección en primer lugar se describe y analiza la especificación de alto nivel del controlador del display de 7 segmentos de 8 dígitos utilizado en el Sistema RVfpga, y a continuación se proponen ejercicios para su uso.

A. Especificación de alto nivel

El controlador del display de 7 segmentos de 8 dígitos usado en este curso ha sido diseñado a medida para el Sistema RVfpga. Incluye dos registros, llamados *Enables_Reg* y *Digits_Reg*, que están mapeados a las direcciones 0x80001038 y 0x8000103C respectivamente (nótese que estas direcciones son direcciones no utilizadas dentro del rango de direcciones reservadas para el Controlador del Sistema, que se puede ver en <https://github.com/chipsalliance/Cores-SweRVolf>).

TAREA: Localizar la declaración de los registros *Enables_Reg* y *Digits_Reg*, así como el lugar donde se les asigna un valor. El display de 7 segmentos de 8 dígitos se implementa en el archivo:

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

Enables_Reg es un registro de 8 bits en el que cada bit determina si el dígito correspondiente está en *ON* (0) o en *OFF* (1). *Digits_Reg* es un registro de 32 bits en el que cada grupo de 4 bits representa el valor hexadecimal a mostrar en el dígito correspondiente. Por ejemplo, para mostrar 71 en los dos dígitos de la derecha, el programador asignaría los siguientes valores a los registros:

- *Enables_Reg* = 0xFC (dos dígitos de la derecha activados)
- *Digits_Reg* = 0x00000071 (valor = 71)

4. EJERCICIOS BÁSICOS

Ejercicio 1. Escriba un programa en ensamblador de RISC-V y un programa en C que muestre el valor de los interruptores en los cuatro dígitos de la derecha de los displays de 7 segmentos.

Ejercicio 2. Escriba un programa en ensamblador de RISC-V y un programa en C que muestre la cadena "0-1-2-3-4-5-6-7-8" moviéndose de derecha a izquierda en el display de 7 segmentos de 8 dígitos. Es decir, en primer lugar únicamente debería aparecer el 0 en el dígito más a la derecha. Tras ello debería moverse a la izquierda y el 1 debería aparecer en el dígito más a la derecha, y así sucesivamente.

5. CONTROLADOR DEL DISPLAY DE 7 SEGMENTOS DE 8 DÍGITOS: IMPLEMENTACIÓN DE BAJO NIVEL Y SIMULACIÓN

Hasta el momento sólo se ha visto cómo usar el display de 7 segmentos de 8 dígitos. En esta sección se describe su implementación de bajo nivel y se analiza su simulación en RVfpgaSim al ejecutar un ejemplo sencillo en ensamblador. Por último, se proporcionan ejercicios para modificar el controlador del display de 7 segmentos de 8 dígitos.

A. Implementación de bajo nivel del controlador del display de 7 segmentos de 8 dígitos

Al igual que en las anteriores prácticas de Entrada/Salida de propósito general (GPIO), se divide en tres fases el análisis del controlador del display de 7 segmentos de 8 dígitos:

1. Conexión entre el SoC y el dispositivo de Entrada/Salida de la placa (región sombreada a la izquierda en la Figura 4);
2. Integración del nuevo controlador, que se incluye dentro del Controlador del Sistema SweRVolfX contenido en el SoC (región sombreada central en la Figura 4);
3. Conexión entre el nuevo controlador y el core SweRV EH1 (región sombreada a la derecha en la Figura 4).

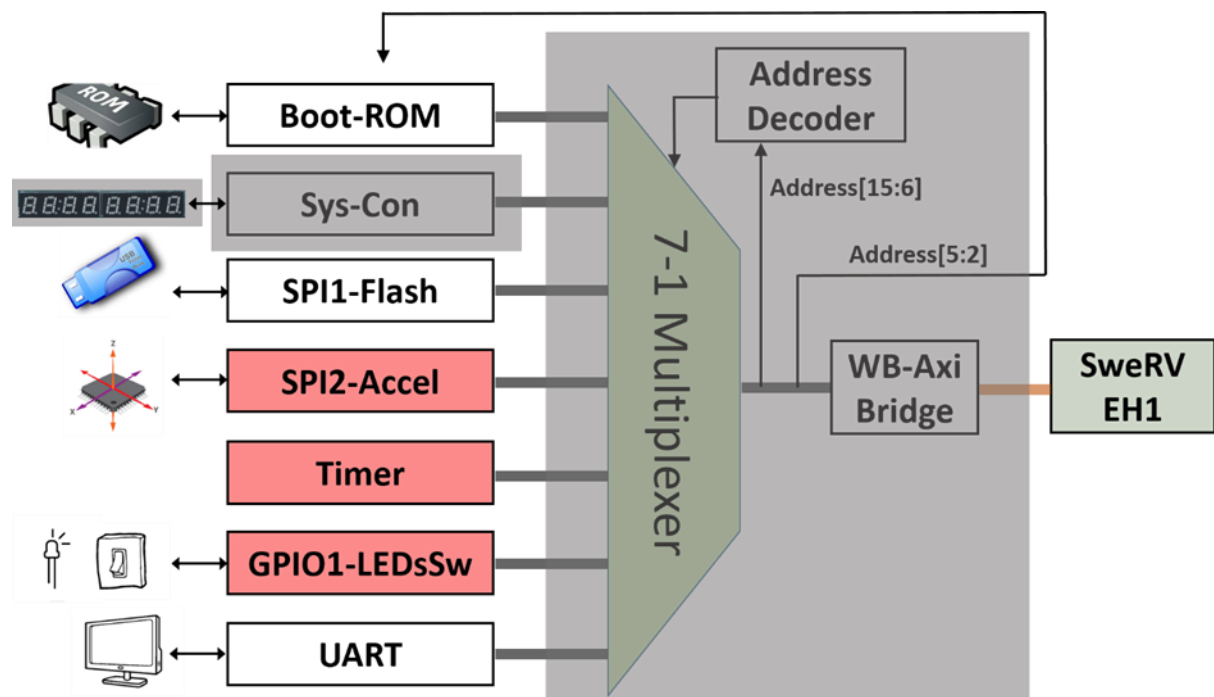


Figura 4. Análisis en 3 fases del controlador del display de 7 segmentos de 8 dígitos

1. Conexión de los LEDs/Interruptores al SoC

El archivo de restricciones del proyecto (*[RVfpgaPath]/RVfpga/src/rvfpganexys.xdc*) define la conexión entre las señales de entrada/salida del SoC y la placa. Cada dispositivo de Entrada/Salida de la placa FPGA Nexys A7 está conectado a un pin específico de la FPGA. La señal que conecta los ocho ánodos (ver Figura 3) se llama *AN[i]* (con *i* que va de 0 a 7), y las señales que conectan los cátodos de segmentos similares en los 8 dígitos (ver Figura 3) se llaman *CA, CB, CC, CD, CE, CF* y *CG*. Figura 5 muestra el fragmento del archivo de restricciones en el que se definen estas conexiones.

```
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]
```

Figura 5. Conexión de las entradas del displays de 7 segmentos de 8 dígitos (archivo *rvfpganexys.xdc*).

En las líneas 50-51 del módulo superior del sistema (módulo *rvfpganexys*, implementado en el archivo *[RVfpgaPath]/RVfpga/src/rvfpganexys.sv*) puede ver las señales de entrada del display de 7 segmentos de 8 dígitos conectadas al SoC, *AN[i]* y *CA...CG* (parte izquierda de la Figura 6), y al final de ese módulo (parte derecha de la Figura 6) se puede ver su conexión con el módulo *swervolf_core* (tenga en cuenta que las señales *CA...CG* se renombran en ese módulo como *Digits_Bits[6:0]*).

```
25 module rvfpganexys
26     #(parameter bootrom_file = "boot_main.mem")
27     (input wire clk,
28      input wire rstn,
29      output wire [12:0] ddram_a,
30      output wire [2:0] ddram_ba,
31      output wire ddram_ras_n,
32      output wire ddram_cas_n,
33      output wire ddram_we_n,
34      output wire ddram_cs_n,
35      output wire [1:0] ddram_dm,
36      inout wire [15:0] ddram_dq,
37      inout wire [1:0] ddram_dqs_p,
38      inout wire [1:0] ddram_dqs_n,
39      output wire ddram_clk_p,
40      output wire ddram_clk_n,
41      output wire ddram_cke,
42      output wire ddram_odt,
43      output wire o_flash_cs_n,
44      output wire o_flash_mosi,
45      input wire i_flash_miso,
46      input wire i_uart_rx,
47      output wire o_uart_tx,
48      inout wire [15:0] i_sw,
49      output reg [15:0] o_led,
50      output reg [7:0] AN,
51      output reg CA, CB, CC, CD, CE, CF, CG,
```

```
256 .i ram_init_error (litedram_init_error),
257 .io data ((i_sw[15:0], qpio out[15:0])),
258 .AN (AN),
259 .Digits_Bits ({CA, CB, CC, CD, CE, CF, CG}),
260 .o accel_sclk (accel_sclk),
```

Figura 6. Conexión del display de 7 segmentos de 8 dígitos al SoC (archivo: *rvfpganexys.sv*).

Finalmente, las dos señales se insertan desde el módulo **swervolf_core** en el módulo del Controlador del Sistema (**swervolf_syscon**) (véase la Figura 7), en donde se implementa el controlador del display de 7 segmentos de 8 dígitos.

```

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_adr        (wb_m2s_sys_adr[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. Conexión del display de 7 segmentos de 8 dígitos al Controlador del Sistema (archivo: swervolf_core.v).

TAREA: Siga las señales CA-CG y AN desde el archivo de restricciones hasta el módulo del Controlador del Sistema (en donde CA-CG se fusionan en el array *Digits_Bits*). Necesitará examinar los siguientes archivos:

```

[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. Integración del controlador del display de 7 segmentos de 8 dígitos en el SoC

En las líneas 276-288 del módulo **swervolf_syscon** ([RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf_syscon.v) se instancia el controlador del display de 7 segmentos de 8 dígitos y se integra en el SoC (véase 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. Instanciación del controlador de 7 segmentos de 8 dígitos (archivo: *swervolf_syscon.v*).

El módulo **SevSegdisplays_Controller** recibe, además de la señal de reloj (*i_clk*, renombrada como *clk*) y la señal de reset (*i_rst*, renombrada como *rst_n*), dos señales de entrada (*Enables_Reg* y *Digits_Reg*), que son los dos registros de control mapeados en memoria ya descritos. Este módulo genera dos señales, *AN* y *Digits_Bits*, que se conectan a los displays de 7 segmentos de la placa. Para el ejemplo que muestra 71 en los dos dígitos de la derecha, el **SevSegdisplays_Controller** asignaría los siguientes valores a las señales *AN* y *Digits_Bits*:

- De 0 a 2ms: La señal *AN[0]* está en bajo para permitir que el dígito 0 (el más a la derecha) muestre un valor. Las señales *Digits_Bits[5]* y *Digits_Bits[4]* (que corresponden a *CB* y *CC*) también se ponen en bajo para mostrar "1" en el dígito 0 (el dígito más a la derecha). Todas las demás señales están en alto.
- De 2 a 4ms: La señal *AN[1]* es puesta en bajo para permitir que el dígito 1 muestre un valor. *Digits_Bits[6]*, *Digits_Bits[5]* y *Digits_Bits[4]* (que corresponden a *CA*, *CB*, y *CC*) son puestos en alto para mostrar "7" en el dígito 1. Todas las demás señales son puestas en alto.
- De 4 a 16ms: *AN[2]...AN[7]* son puestas en alto en intervalos de 2 ms para que no muestren valores. Los segmentos también se ponen en alto para los dígitos restantes, los dígitos 2-7.

El módulo **SevSegdisplays_Controller** está implementado en las líneas 295-366 del archivo

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

Contiene las siguientes subunidades:

- Dos multiplexores que cada 2ms seleccionan el valor a enviar a las señales *AN* y *Digits_Bits*. El multiplexor se implementa dentro del módulo **SevSegMux**.
- Para crear el período de 2ms, se utiliza un módulo **contador** provisto en los archivos *counter.sv* y *delta_counter.sv*, ambos incluidos en la carpeta *[RVfpgaPath]/RVfpga/src/OtherSources/pulp-platform.org__common_cells_1.20.0/src*. El contador está configurado para contar de 0 a 2^{19} , y los 3 bits más significativos, que cambian aproximadamente cada 2ms, se utilizan como señales de selección para los dos multiplexores descritos anteriormente.
- En el módulo **SevenSegDecoder** se implementa un decodificador, que genera los valores de los segmentos para un valor hexadecimal de 4 bits dado.

TAREAS: Analizar el módulo **SevSegdisplays_Controller** en detalle. La simulación que se lleva a cabo en la siguiente sección puede ayudarle en esta tarea. También puede ampliar la simulación añadiendo nuevas señales si es necesario.

3. Conexión entre el controlador del display de 7 segmentos de 8 dígitos y el core SweRV EH1

Como se describe en la Práctica 6, los controladores de dispositivo se conectan al core SweRV EH1 mediante un multiplexor (véase Figura 4). Recuerde que el multiplexor 7:1 (Figura 9) se implementa en el archivo

[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Interconnect/WishboneInterconnect/wb_intercon.v, que se instancia en las líneas 104-205 del archivo

[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Interconnect/WishboneInterconnect/wb_intercon.vh. Este último archivo se incluye en la línea 168 del módulo **swervolf_core** que se encuentra aquí: *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/swervolf_core.v*.

El multiplexor selecciona qué periférico leer o escribir, conectando la CPU (señales *wb_io_** - líneas 115-126 de la Figura 9) con el Bus Wishbone de un periférico (líneas 127-138 de la Figura 9), dependiendo de la dirección (líneas 110-111). Por ejemplo, si la dirección generada por la CPU está en el rango 0x80001000-0x8000103F, se selecciona el Controlador del Sistema, y por lo tanto las señales *wb_io_** se conectarán con las señales *wb_sys_**.

```

108 wb_mux
109 #(.num_slaves (7),
110 .MATCH_ADDR ({32'h00000000, 32'h00001000, 32'h00001040, 32'h00001100, 32'h00001200, 32'h00001400, 32'h00002000}),
111 .MATCH_MASK ({32'hffff0000, 32'hfffffc00, 32'hfffffc00, 32'hfffffc00, 32'hfffffc00, 32'hfffffc00, 32'hffff0000}))
112 wb_mux_io
113 (.wb_clk_i (wb_clk_i),
114 .wb_rst_i (wb_rst_i),
115 .wbm_adr_i (wb_io_adr_i),
116 .wbm_dat_i (wb_io_dat_i),
117 .wbm_sel_i (wb_io_sel_i),
118 .wbm_we_i (wb_io_we_i),
119 .wbm_cyc_i (wb_io_cyc_i),
120 .wbm_stb_i (wb_io_stb_i),
121 .wbm_cti_i (wb_io_cti_i),
122 .wbm_bte_i (wb_io_bte_i),
123 .wbm_dat_o (wb_io_dat_o),
124 .wbm_ack_o (wb_io_ack_o),
125 .wbm_err_o (wb_io_err_o),
126 .wbm_rty_o (wb_io_rty_o),
127 .wbs_adr_o ({wb_rom_adr_o, wb_sys_adr_o, wb_spi_flash_adr_o, wb_spi_accel_adr_o, wb_ptc_adr_o, wb_gpio_adr_o, wb_uart_adr_o}),
128 .wbs_dat_o ({wb_rom_dat_o, wb_sys_dat_o, wb_spi_flash_dat_o, wb_spi_accel_dat_o, wb_ptc_dat_o, wb_gpio_dat_o, wb_uart_dat_o}),
129 .wbs_sel_o ({wb_rom_sel_o, wb_sys_sel_o, wb_spi_flash_sel_o, wb_spi_accel_sel_o, wb_ptc_sel_o, wb_gpio_sel_o, wb_uart_sel_o}),
130 .wbs_we_o ({wb_rom_we_o, wb_sys_we_o, wb_spi_flash_we_o, wb_spi_accel_we_o, wb_ptc_we_o, wb_gpio_we_o, wb_uart_we_o}),
131 .wbs_cyc_o ({wb_rom_cyc_o, wb_sys_cyc_o, wb_spi_flash_cyc_o, wb_spi_accel_cyc_o, wb_ptc_cyc_o, wb_gpio_cyc_o, wb_uart_cyc_o}),
132 .wbs_stb_o ({wb_rom_stb_o, wb_sys_stb_o, wb_spi_flash_stb_o, wb_spi_accel_stb_o, wb_ptc_stb_o, wb_gpio_stb_o, wb_uart_stb_o}),
133 .wbs_cti_o ({wb_rom_cti_o, wb_sys_cti_o, wb_spi_flash_cti_o, wb_spi_accel_cti_o, wb_ptc_cti_o, wb_gpio_cti_o, wb_uart_cti_o}),
134 .wbs_bte_o ({wb_rom_bte_o, wb_sys_bte_o, wb_spi_flash_bte_o, wb_spi_accel_bte_o, wb_ptc_bte_o, wb_gpio_bte_o, wb_uart_bte_o}),
135 .wbs_dat_i ({wb_rom_dat_i, wb_sys_dat_i, wb_spi_flash_dat_i, wb_spi_accel_dat_i, wb_ptc_dat_i, wb_gpio_dat_i, wb_uart_dat_i}),
136 .wbs_ack_i ({wb_rom_ack_i, wb_sys_ack_i, wb_spi_flash_ack_i, wb_spi_accel_ack_i, wb_ptc_ack_i, wb_gpio_ack_i, wb_uart_ack_i}),
137 .wbs_err_i ({wb_rom_err_i, wb_sys_err_i, wb_spi_flash_err_i, wb_spi_accel_err_i, wb_ptc_err_i, wb_gpio_err_i, wb_uart_err_i}),
138 .wbs_rty_i ({wb_rom_rty_i, wb_sys_rty_i, wb_spi_flash_rty_i, wb_spi_accel_rty_i, wb_ptc_rty_i, wb_gpio_rty_i, wb_uart_rty_i});
139
140 endmodule

```

CPU/Controller Signals

Peripheral Signals

Figura 9. Multiplexor 7-1 que selecciona el periférico conectado con la CPU (archivo: *wb_intercon.v*).

Los registros incluidos en el Controlador del Sistema se escriben desde la CPU conectándolos directamente a la señal de datos del Bus Wishbone (*i_wb_dat*), en base a la dirección (*i_wb_adr*) generada por la CPU (líneas 162-228 del módulo *swervolf_syscon*).

TAREA: Examine las líneas 162-228 del módulo *swervolf_syscon* para entender cómo se asignan las direcciones en el Controlador del Sistema. Céntrese en las líneas 219 a 227 (Figura 10), que se refieren a los registros *Enables_Reg* y *Digits_Reg* (como se mencionó antes, las direcciones asignadas a estos dos registros son 0x80001038 y 0x8000103C respectivamente).

```

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

```

Figura 10. Conexión entre el display de 7 segmentos de 8 dígitos y el Core (archivo *swervolf_syscon.v*).

B. Simulación en Verilator

En esta sección se utiliza **RVfpgaSim** para examinar las principales señales del controlador del display de 7 segmentos de 8 dígitos al ejecutar el procesador un ejemplo simple que

utiliza este periférico. En la simulación, se analizan las señales *AN* y *Digits_Bits* mientras se ejecuta el ejemplo de la Figura 11, que escribe 71 en los dos dígitos de la derecha. Puede encontrar este programa en: [RVfpgaPath]/RVfpga/Labs/Lab7/71_7SegDispl (también puede encontrar la versión C en: [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
```

Figura 11. Ejemplo 71_7SegDispl.S

La Figura 12 muestra la versión desensamblada del programa 71_7SegDispl. elf, que, después la compilación en PlatformIO, se puede encontrar en: [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. Versión desensamblada del ejemplo 71_7SegDispl.S

Siga los siguientes pasos para ejecutar la simulación. (Si prefiere no ejecutar la simulación, puede ir directamente al paso 7.)

1. En este caso, sólo para la simulación, debe reducir el período de reloj cambiando COUNT_MAX (véase la línea 295 del archivo [RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/SystemController/swervolf_syscon.v) de 20 a 5; de lo contrario llevaría demasiado tiempo visualizar los resultados. Modifique el valor de COUNT_MAX y luego recompile RVfpgaSim ejecutando los siguientes comandos (esto se explicó en el Guía de Inicio):


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

Se debe generar un nuevo archivo *Vrvfpgasim* (el archivo binario de simulación de RVfpgaSim), dentro del directorio *[RVfpgaPath]/RVfpga/verilatorSIM*.

WINDOWS: en caso de que esté usando Windows, tiene que ejecutar estos comandos dentro de la terminal Cygwin (consulte la Sección 6 y el Apéndice C de la Guía de Inicio para instrucciones detalladas). Tenga en cuenta que la carpeta *C:\Windows* se puede encontrar dentro de Cygwin en: */cygdrive/c*.

MacOS: Consulte el Apéndice D de la Guía de Inicio para obtener instrucciones detalladas.

2. Abra VSCode/PlataformalO en su computadora.
3. En la barra superior, haga clic en *File - Open Folder...*, y busque la carpeta *[RVfpgaPath]/RVfpga/Labs/Lab7*
4. Seleccione la carpeta *71_7SegDispl* (no la abra, sólo selecciónela) y haga clic en OK. El ejemplo se abrirá en PlatformIO.
5. Abra el archivo *platformio.ini* y compruebe si la ruta del archivo binario de simulación RVfpgaSim es correcta. Recuerde de la Guía de Inicio que debería ser:


```
board_debug.verilator.binary =
[RVfpgaPath]/RVfpga/verilatorSIM/Vrvfpgasim
```
6. Ejecute la simulación haciendo clic en el icono de PlatformIO en la barra del menú de la izquierda , luego expanda las Project Tasks → env:swervolf_nexys → Platform y haga clic en Generate Trace.

Se debería haber generado el archivo *trace.vcd* dentro de *[RVfpgaPath]/RVfpga/Labs/Lab7/71_7SegDispl/.pio/build/swervolf_nexys*, que se puede abrir con *GTKWave* ejecutando el siguiente comando:

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

WINDOWS: la carpeta *gtkwave64* que descargó, incluye una aplicación llamada *gtkwave.exe* dentro de la carpeta *bin*. Ejecute *GTKWave* haciendo doble clic en esa aplicación. En la parte superior de la aplicación, haga clic en **File – Open New Tab**, y abra el archivo *trace.vcd* generado en la carpeta *[RVfpgaPath]/RVfpga/Labs/Lab7/71_7SegDispl/.pio/build/swervolf_nexys*.

7. Incluya las siguientes señales en la simulación (acceda a los módulos indicados para localizar cada una de las señales):
 - rvfpgasim - swervolf - syscon - SegDispl_Ctr
 - ✓ Señales de entrada: **Enables_Reg** y **Digits_Reg**.
 - ✓ Señales de salida: **AN** y **Digits_Bits**.
8. Analice la simulación que se muestra en la Figura 13. Inicialmente, los valores mostrados en los ocho displays de 7 segmentos son todos 0 (inicialmente todos los dígitos están habilitados dado que *Enables_Reg=0*). Luego se deshabilitan los seis dígitos más a la izquierda al escribir *0xFC* en *Enables_Reg* (instrucción *sb* en la Figura 12) y se escribe *71* en los dos dígitos más a la derecha escribiendo *0x71* en *Digits_Reg* (instrucción *sw* en la Figura 12). El efecto sobre las señales de salida es el siguiente

(como se muestra en la Figura 13):

- En el primer período: $AN=0xFE$ y $Digits_Bits=0x4F$, mostrando así 1 en el dígito más a la derecha, el dígito 0.
- En el segundo período: $AN=0xFD$ y $Digits_Bits=0x0F$, mostrando así 7 en el siguiente dígito, el dígito 1.
- En los siguientes seis períodos: $AN=0xFF$ y $Digits_Bits=0x01$, apagando así los seis dígitos más a la izquierda.
- Este proceso se repite.

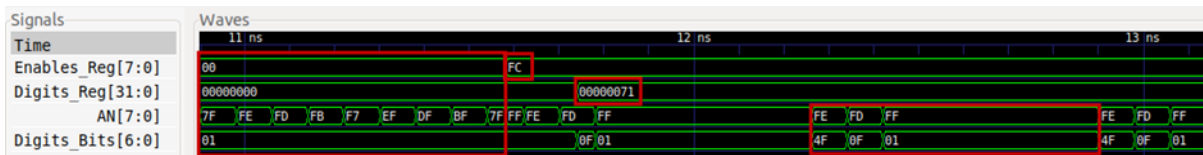


Figura 13. Escritura del valor 71 en los dos dígitos de la derecha del display de 7 segmentos de 8 dígitos

9. Antes de continuar, no olvide restaurar el valor de `COUNT_MAX` a su valor original (`COUNT_MAX=20`).

6. EJERCICIOS AVANZADOS

Ejercicio 3. Modifique el controlador descrito en esta práctica para que el display de 7 segmentos de 8 dígitos pueda mostrar cualquier combinación de LEDs encendidos/apagados.

- Ahora no se necesita un registro de habilitación. En cambio, necesita ocho registros de 7 bits, que debe nombrar así: `Segments_Digit0` - `Segments_Digit7`, uno para cada uno de los ocho displays de 7 segmentos. En cada uno de estos registros, cada bit indica si el segmento correspondiente está ON (0) o OFF (1). Por ejemplo, si todos los bits del primer registro (`Segments_Digit0`) son 0, todos los segmentos del dígito de más a la derecha estarán en encendidos, mientras que si todos los bits del primer registro son 1, todos los segmentos del dígito de más a la derecha estarán en apagados.
- Puede mapear estos dos nuevos registros a las mismas direcciones que se usaron antes (primero elimine los dos registros anteriores `Enables_Reg` y `Digits_Reg`):
 - `Segments_Digit0` \leftrightarrow Dirección `0x80001038`
 - `Segments_Digit1` \leftrightarrow Dirección `0x80001039`
 - ...
 - `Segments_Digit7` \leftrightarrow Dirección `0x8000103F`
- Tenga en cuenta que ya no necesita el decodificador 4-7 (módulo **SevenSegDecoder**), ya que la información proporcionada por el programa ya está decodificada.

Ejercicio 4. Utilice el nuevo controlador para imprimir lo siguiente en el display de 7 segmentos de 8 dígitos: " I SAY HI". Como de costumbre, implemente tanto la versión en ensamblador de RISC-V como la versión en C del programa.