



IMAGINATION大學計劃

# RVfpga實驗8

## 計時器

## 1. 簡介

硬體計時器是微控制器和SoC中常見的週邊設備，通常用於產生精確時序。計時器以固定頻率（該頻率通常是可配置的）遞增或遞減計數器，然後在計數器達到零或預定義值時中斷處理器。更複雜的計時器還可以執行其他功能，例如產生脈寬調變（Pulse-Width Modulated，PWM）波形以控制馬達轉速或燈光亮度。

本實驗的結構編排與之前的實驗類似，我們首先介紹RVfpga系統中所含計時器的高階規格，然後解釋說明其低階實作。此外，還提供了基礎練習和進階練習，以展示如何使用和修改計時器。

## 2. RVfpga系統中所含計時器的高階規格

在本節中，我們首先分析RVfpga系統中使用的計時器的高階規格，然後提供一個使用該週邊設備的練習。

### A. 計時器高階規格

已從OpenCores（<https://opencores.org/projects/ptc>）取得RVfpga系統中使用的計時器模組。如果下載套件，其中會隨附一個文件，用於描述該模組的高階規格（此文件位於以下位置：*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/ptc/docs/ptc\_spec.pdf*）。我們在此總結了計時器模組的主要操作和特性；有關完整的資訊，請參閱上述文件。

計時器模組的主要特性如下：

- 使用Wishbone互連
- 32位元計數器/計時器模式
- 單次執行或連續執行PWM/計時器/計數器（PWM/Timer/Counter，PTC）
- 可程式化PWM（脈寬調變）模式
- 系統時鐘和外部時鐘來源，用於計時器功能
- 高電平/低電平參考和擷取暫存器
- PWM輸出驅動器的三態控制
- PTC功能可向CPU發出中斷

計時器模組規格文件的第4節介紹了計時器模組內部可用的控制和狀態暫存器，每個暫存器都分配到不同的位址（請參閱表1）。RVfpga系統中計時器的基本位址為**0x80001200**。

**表1. 計時器暫存器**

名稱	位址	寬度	存取	說明
RPTC_CNTR	0x80001200	1-32	R/W	主PTC計數器
RPTC_HRC	0x80001204	1-32	R/W	PTC高電平參考/擷取暫存器
RPTC_LRC	0x80001208	1-32	R/W	PTC低電平參考/擷取暫存器
RPTC_CTRL	0x8000120C	9	R/W	控制暫存器

RPTC\_CNTR暫存器是實際的計數器暫存器，每個計數器/計時器每個時鐘週期遞增一次。RPTC\_CTRL暫存器用於控制計時器模組；表2顯示了其中每個位元的功能。RPTC\_HRC和RPTC\_LRC用作參考/擷取暫存器。

**表2. RPTC\_CTRL位元**

位元	存取	重設	名稱和說明
0	R/W	0	<b>EN</b> 設為1時，RPTC_CNTR遞增。
1	R/W	0	<b>ECLK</b> 選擇時鐘訊號：外部時鐘（通過 <code>ptc_ecgt</code> ）（1）或系統時鐘（0）。
2	R/W	0	<b>NEC</b> 用於選擇外部時脈（ <code>ptc_ecgt</code> ）的負邊緣/正邊緣和低電平/高電平週期。
3	R/W	0	<b>OE</b> 啟用PWM輸出驅動器。
4	R/W	0	<b>SINGLE</b> 設定時，RPTC_CNTR在達到RPTC_LRC值後不遞增。清零時，RPTC_CNTR在達到RPTC_LCR暫存器中的值後重新啟動。
5	R/W	0	<b>INTE</b> 設為1時，當RPTC_CNTR值等於RPTC_LRC或RPTC_HRC的值時，PTC會將中斷置為有效。清除訊號時，中斷將被屏蔽。
6	R/W	0	<b>INT</b> 讀取時，該位元表示待處理的中斷。設為1時，表示有一個中斷待處理。當該位元寫入1時，中斷請求將被清除。
7	R/W	0	<b>CNTRRST</b> 設為1時，將重設RPTC_CNTR。清零時，計數器將正常工作。
8	R/W	0	<b>CAPTE</b> 設為1時，RPTC_CNTR將被擷取到RPTC_LRC或RPTC_HRC暫存器中。清零時，擷取功能將被屏蔽。

**任務：**在計時器模組中尋找暫存器RPTC\_CNTR、RPTC\_HRC、RPTC\_LRC和RPTC\_CTRL的宣告，以及這些暫存器的位址定義（分別為0x80001200、0x80001204、0x80001208和0x8000120C）。計時器模組位於資料夾  
`[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/ptc`內。

計時器可以在不同的模式下執行（接下來，我們簡要介紹本實驗中將使用的模式；有關詳細資訊，請參閱計時器模組規格文件的第3節）：

- **計時器/計數器模式：**在此模式下，如果啟用了計數器（RPTC\_CTRL[EN] = 1），則系統時鐘或外部時鐘參考會遞增暫存器RPTC\_CNTR。當RPTC\_CNTR等於RPTC\_LRC時，如果設定RPTC\_CTRL[INTE]，則RPTC\_CTRL[INT]變為高電平。
- **PWM模式：**脈寬調變（PWM）訊號是一種使用數字來源產生類比訊號的方法。PWM訊號由兩個定義其行為的值組成：*工作週期*和*頻率*。工作週期描述訊號為高電平的時間，占完成一個週期所用總時間的百分比。頻率是週期重複的頻率。為裝置供電後，如果以足夠快的速率和一定的工作週期迴圈開關數字訊號，輸出會表現為恒壓類比訊號。例如，工作週期為50%（一半的週期時間為高電平）的3.3 V高壓訊號相當於1.67 V（整個週期的平均電壓）的類比負載。相同的訊號，工作週期為33%時則相當於1.1V。要在PWM模式下執行，必須設定RPTC\_CTRL[OE]。暫存器RPTC\_HRC和RPTC\_LRC應分

別設為PWM輸出訊號的高電平週期和低電平週期的值：（RPTC\_CNTR）重設後，PWM訊號應變為高電平RPTC\_HRC時鐘週期；（RPTC\_CNTR）重設後，PWM訊號應變為低電平RPTC\_LRC時鐘週期。

### 3. 基本練習

**練習1.** 編寫一個程式，在8位7段顯示器上顯示升序計數。該值應大約每秒鐘改變一次，一秒延遲需使用計時器模組產生。

- a. 首先，用RISC-V組合語言編寫程式並在Nexys A7開發板上執行該程式。
- b. 然後，使用相同程式在Verilator中進行模擬。可新增以下訊號：系統時鐘、用於儲存在8位7段顯示器中顯示的值的處理器暫存器，以及計時器暫存器RPTC\_CNTR、RPTC\_LRC、RPTC\_HRC和RPTC\_CRTL。
- c. 現在，用C語言編寫程式並在Nexys A7開發板上執行該程式。
- d. 與RISC-V組合語言程式的（b）部分一樣，在Verilator中模擬您的C程式。

### 4. 計時器低階實作

在本節中，我們首先介紹RVfpga系統中計時器模組的低階實作，然後提供一些練習。在練習中，我們將首先修改該模組，然後在程式中使用該模組來控制Nexys A7開發板上的三色LED。

#### A. 計時器的低階實作

與先前實驗中遵循的方案類似，我們分階段來進行計時器模組分析。

1. 在SweRVofX SoC中整合新模組（圖1中左側陰影區域）
2. 新模組與SweRV EH1核心之間的連接（圖1中右側陰影區域）。

請注意，此週邊設備（計時器）未物理連接到Nexys A7開發板，這一點與先前的實驗有所不同。計時器在SweRVofX內部。

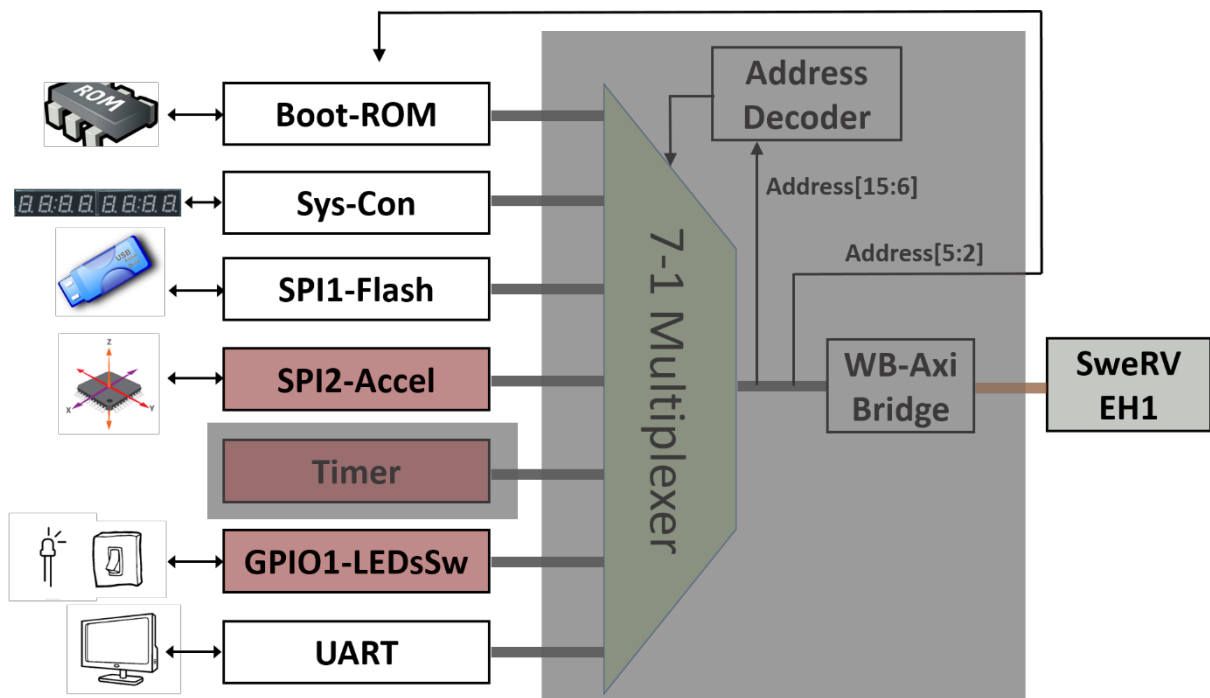


圖1. 計時器模組分析分2個階段

#### i. 計時器模組在SoC中的整合

在模組 **swervolf\_core** (`[RVfpgaPath]/RVfpga/src/SweRVolfSoC/swervolf_core.v`) 的第361-379行，計時器模組實例化（請參閱圖2）。

```

358 // PTC
359 wire      ptc_irq;
360
361 ptc_top timer_ptc(
362     .wb_clk_i      (clk),
363     .wb_rst_i      (wb_rst),
364     .wb_cyc_i      (wb_m2s_ptc_cyc),
365     .wb_adr_i      ({2'b0,wb_m2s_ptc_adr[5:2],2'b0}),
366     .wb_dat_i      (wb_m2s_ptc_dat),
367     .wb_sel_i      (4'b1111),
368     .wb_we_i      (wb_m2s_ptc_we),
369     .wb_stb_i      (wb_m2s_ptc_stb),
370     .wb_dat_o      (wb_s2m_ptc_dat),
371     .wb_ack_o      (wb_s2m_ptc_ack),
372     .wb_err_o      (wb_s2m_ptc_err),
373     .wb_inta_o     (ptc_irq),
374     // External PTC Interface
375     .gate_clk_pad_i (),
376     .capt_pad_i    (),
377     .pwm_pad_o     (),
378     .oen_padoen_o  ()
379 );

```

圖2. 計時器模組的整合（**swervolf\_core.v**檔）。

模組的介面依舊可以分為兩個模組：Wishbone訊號（表3）和外部I/O訊號（表4）。Wishbone訊號允許SweRV EH1核心使用控制器/週邊設備模型與計時器進行通訊。外部I/O訊號將計時器

模組與外部裝置連接；例如，在上述PWM模式下執行時，*pwm\_pad\_o*會提供PWM輸出訊號（在練習2中需使用此訊號將計時器模組與三色LED連接）。

**表3. Wishbone訊號**

連接埠	寬度	方向	說明
<i>wb_cyc_i</i>	1	輸入	指示有效的匯流排週期（核心選擇）
<i>wb_adr_i</i>	15	輸入	位址輸入
<i>wb_dat_i</i>	32	輸入	資料輸入
<i>wb_dat_o</i>	32	輸出	資料輸出
<i>wb_sel_i</i>	4	輸入	指示資料匯流排上的有效位元組（在有效週期內，此訊號必須為0xf）
<i>wb_ack_o</i>	1	輸出	認可輸出（指示正常交易終止）
<i>wb_err_o</i>	1	輸出	錯誤認可輸出（指示異常交易終止）
<i>wb_rty_o</i>	1	輸出	未使用
<i>wb_we_i</i>	1	輸入	置為高電平時寫入交易
<i>wb_stb_i</i>	1	輸入	指示有效的資料傳輸週期
<i>wb_inta_o</i>	1	輸出	中斷輸出

**表4. 外部I/O訊號**

連接埠	寬度	方向	說明
<i>gate_clk_pad_i</i>	1	輸入	外部時鐘/閘門輸入
<i>capt_pad_i</i>	1	輸入	擷取輸入
<i>pwm_pad_o</i>	1	輸出	PWM輸出
<i>oen_padoen_o</i>	1	輸出	PWM輸出驅動器啟用（用於三態或漏極開路驅動器）

如圖2的第365行所示，Wishbone匯流排訊號中由核心提供的位址的位元[5:2]

（*wb\_m2s\_ptc\_adr[5:2]*）用於從4個可用的暫存器中選擇1個（記憶體映射I/O）。因此，我們可以存取位址0x80001200處的暫存器RPTC\_CNTR、位址0x80001204處的暫存器RPTC\_HRC、位址0x80001208處的暫存器RPTC\_LRC和位址0x8000120C處的暫存器RPTC\_CTRL。

## ii. 計時器與SweRV EH1核心的連接

如先前的實驗中所述，裝置控制器通過多工器與SweRV EH1核心連接（圖1）。請記住，7:1多工器（圖3）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*。



```

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'hffffff00, 32'hffffffc0, 32'hffffffc0, 32'hffffffc0, 32'hffffffc0, 32'hffffffc0, 32'hffffff00}))
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

圖3. 選擇與CPU連接的週邊設備的7-1多工器（`wb_intercon.v`檔）

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3的第115-126行）與一個週邊設備的Wishbone匯流排（圖3的第127-138行）連接。例如，如果CPU產生的位址在0x80001200-0x8000123F範圍內，則選擇計時器模組，從而將訊號`wb_io_*`與訊號`wb_ptc_*`連接。

## 5. 進階練習

**練習2.** 修改RVfpgaNexys以將計時器的PWM輸出訊號（`pwm_pad_o`）連接到Nexys A7開發板上的兩個三色LED之一。建議將此新功能新增到在實驗6和7中修改並更新後的RVfpgaNexys系統中。

- Digilent提供了以下有關Nexys A7開發板上的三色LED的資訊：  
<https://reference.digilentinc.com/reference/programmable-logic/nexys-a7/reference-manual>
- 綜上所述，開發板包含兩個三色LED。每個三色LED具有三個輸入訊號，這些訊號驅動三個較小的內部LED的陰極：一個紅色、一個藍色和一個綠色。將其中一個陰極驅動為高電平即可使相應的內部LED點亮。三色LED發出何種顏色取決於目前點亮的內部LED組合。例如，將紅色和藍色驅動為高電平會發出紫色。Digilent強烈建議在驅動三色LED時使用脈寬調變（PWM）。將任何輸入驅動為穩定的邏輯「1」會導致LED的亮度異常。通過確保以不超過50%的工作週期來驅動所有三色訊號，可以避免這種情況。此外，使用PWM還可以極大地擴展三色LED的潛在調色盤。分別將每種顏色的工作週期調整到50%和0%之間，會使不同的顏色以不同的強度點亮，從而幾乎可以顯示任何顏色。
- 基於SweRVolfX中已有的計時器模組建立三個新的計時器模組。每種顏色（紅色、藍色和綠色）應由不同的計時器模組驅動，以便可以接收不同的電壓。

- 使用以下位址範圍將每個新計時器的暫存器映射到記憶體：
  - i. 計時器2：0x80001240-0x8000127F
  - ii. 計時器3：0x80001280-0x800012BF
  - iii. 計時器4：0x800012C0-0x800012FF

請注意，在這種情況下，必須將3個新項目新增到選擇週邊設備的多工器（圖1）。

- 在修改限制檔時，必須考慮到3種顏色連接到以下開發板引腳：
  - iv. LED16\_B  $\leftrightarrow$  PIN R12
  - v. LED16\_G  $\leftrightarrow$  PIN M16
  - vi. LED16\_R  $\leftrightarrow$  PIN N15

**練習3.** 使用16個開關提供的值實作一個程式，該程式使用新的週邊設備來控制三色LED。按從最右到最左的順序，依次使用5個開關調整藍色的工作週期，5個開關調整綠色的工作週期，5個開關調整紅色的工作週期。（最左側的開關將不使用。）

- a. 首先，編寫RISC-V組合語言程式。
- b. 然後，編寫C程式。