



THE IMAGINATION UNIVERSITY PROGRAMME

RVfpga Lab 8

Temporizadores

1. INTRODUÇÃO

Os temporizadores em hardware são periféricos comuns encontrados em microcontroladores e SoCs. São normalmente usados para gerar um tempo preciso. Os temporizadores incrementam ou decrementam um contador com uma frequência fixa, que é muitas vezes configurável, e depois interrompem o processador quando o contador atinge zero ou um valor predefinido. Temporizadores mais sofisticados também podem desempenhar outras funções, tais como gerar formas de onda moduladas de largura de pulso (PWM) para controlar a velocidade de um motor ou o brilho de uma luz.

Neste lab, usando uma estrutura semelhante à dos laboratórios anteriores, descrevemos primeiro a especificação de alto nível do temporizador incluída no Sistema RVfpga e depois explicamos a sua implementação de baixo nível. Tanto exercícios elementares como avançados são propostos que mostram como usar e modificar um temporizador.

2. ESPECIFICAÇÃO DE ALTO-NÍVEL DO TEMPORIZADOR INCLUÍDO NO SISTEMA RVfpga

Nesta secção, analisamos primeiro a especificação de alto-nível do temporizador utilizado no Sistema RVfpga e depois propomos um exercício que utiliza este periférico.

A. Especificação de alto nível do temporizador

O módulo temporizador utilizado no Sistema RVfpga foi obtido em OpenCores (<https://opencores.org/projects/ptc>). Ao descarregar o pacote é fornecido um documento que descreve a especificação de alto-nível do módulo (e que nós fornecemos em: *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/ptc/docs/ptc_spec.pdf*). Resumimos aqui a operação principal e as características do módulo temporizador; no entanto, a informação completa pode ser encontrada no documento acima.

O módulo temporizador tem as seguintes características principais:

- Usa uma Interligação Wishbone
- Contador/temporizador de 32 bits
- Execução única ou contínua de PWM/Temporizador/Contador (PTC)
- Modo PWM programável (modulação da largura do pulso)
- Relógio do sistema e fontes de relógio externas para funcionalidade do temporizador
- Registos de referência e captura HI/LO
- Controlo de *tri-state* para o controlador de saída PWM
- As funcionalidades do PTC podem causar uma interrupção no CPU

A secção 4 do documento de especificação do módulo do temporizador descreve os registos de controlo e estado disponíveis dentro do módulo temporizador, cada um dos quais é atribuído a um endereço diferente (ver Tabela 1). O endereço base do temporizador no Sistema RVfpga é **0x80001200**.

Tabela 1. Registos do temporizador

Nome	Endereço	Largura	Acesso	Descrição
RPTC_CNTR	0x80001200	1-32	R/W	Contador PTC principal
RPTC_HRC	0x80001204	1-32	R/W	Registo de referência/captura PTC HI
RPTC_LRC	0x80001208	1-32	R/W	Registo de referência/captura PTC LO
RPTC_CTRL	0x8000120C	9	R/W	Registo de controlo

O registo RPTC_CNTR é o verdadeiro registo do contador, e é incrementado em cada ciclo de relógio do contador/temporizador. O registo RPTC_CTRL é utilizado para controlar o módulo temporizador; A Tabela 2 mostra a função de cada uma das suas partes. RPTC_HRC e RPTC_LRC são utilizados como registos de referência/captura.

Tabela 2. Bits do RPTC_CTRL

Bit	Acesso	Reset	Nome & Descrição
0	R/W	0	EN Quando é 1, o RPTC_CNTR incrementa.
1	R/W	0	ECLK Seleciona o sinal do relógio: relógio externo, através de <i>ptc_ecgt</i> ('1') ou relógio do sistema ('0').
2	R/W	0	NEC Utilizado para selecionar a transição ascendente/descendente e o nível baixo/alto do relógio externo (<i>ptc_ecgt</i>).
3	R/W	0	OE Ativa o controlador de saída PWM.
4	R/W	0	SINGLE Quando é '1', RPTC_CNTR não é incrementado depois atinge valor igual ao valor RPTC_LRC. Quando é 0, RPTC_CNTR é reiniciado depois de atingir o valor no registo RPTC_LCR.
5	R/W	0	INTE Quando é '1', o PTC ativa a interrupção quando o valor de RPTC_CNTR for igual ao valor de RPTC_LRC ou RPTC_HRC. Quando é '0', as interrupções são mascaradas.
6	R/W	0	INT Quando lido, este bit representa uma interrupção pendente. Quando é '1', uma interrupção está pendente. Quando este bit é escrito com '1', o pedido de interrupção é apagado.
7	R/W	0	CNTRRST Quando é ativo, RPTC_CNTR é reposto. Quando desativado, ocorre o funcionamento normal do contador.
8	R/W	0	CAPTE Quando definido, RPTC_CNTR é capturado nos registos RPTC_LRC ou RPTC_HRC. Quando desativado, a função de captura é mascarada.

TAREFA: Localize a declaração dos registos RPTC_CNTR, RPTC_HRC, RPTC_LRC e RPTC_CTRL no módulo temporizador, bem como na definição dos seus endereços (0x80001200, 0x80001204, 0x80001208 e 0x8000120C respectivamente). O módulo temporizador está disponível dentro da pasta *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/ptc*.

O temporizador pode funcionar em diferentes modos (descrevemos brevemente os modos que utilizará neste laboratório; consulte a Secção 3 do documento de especificação do módulo temporizador para obter mais detalhes):

- **Modo temporizador/contador:** Neste modo, o relógio do sistema ou os incrementos de referência do relógio externo registam RPTC_CNTR se o contador estiver activado (RPTC_CTRL[EN]=1). Quando RPTC_CNTR é igual ao RPTC_LRC, se RPTC_CTRL[INTE] está definido, RPTC_CTRL[INT] fica ativo.
- **Modo PWM:** Um sinal de modulação da largura de pulso / *Pulse Width Modulation* (PWM) é um método para gerar um sinal analógico usando uma fonte digital. Um sinal

PWM consiste em dois valores que definem o seu comportamento: o *ciclo de funcionamento* e a *frequência*. O ciclo de funcionamento descreve a quantidade de tempo que o sinal é ativo como uma percentagem do tempo total que leva a completar um ciclo. A frequência é a taxa de repetição desse ciclo. Ao fazer o ciclo de um sinal digital desligado e ligado a uma velocidade suficientemente rápida, e com um determinado ciclo de funcionamento, a saída parecerá comportar-se como um sinal analógico de tensão constante ao fornecer energia aos dispositivos. Por exemplo, um sinal com um ciclo de funcionamento de 50% (metade do tempo de ciclo é elevado) e uma tensão elevada de 3,3 V pareceria a uma carga analógica como 1,67 V (a tensão média ao longo do ciclo). O mesmo sinal com um ciclo de funcionamento de 33% pareceria ser 1,1 V. Para operar no modo PWM, RPTC_CTRL[OE] deve ser definido. Os registos RPTC_HRC e RPTC_LRC devem ser definidos com o valor dos períodos alto e baixo do sinal de saída PWM: o sinal PWM deve passar a nível alto RPTC_HRC ciclos de relógio após a (re)iniciação (do RPTC_CNTR); e o sinal PWM deve passar a nível baixo RPTC_LRC ciclos de relógio após (re)iniciação (do RPTC_CNTR).

3. EXERCÍCIO ELEMENTAR

Exercício 1. Escrever um programa que mostre uma contagem ascendente no mostrador de 8 dígitos de 7 segmentos. O valor deve mudar cerca de uma vez por segundo e, para criar este atraso, é necessário utilizar o módulo temporizador.

- Primeiro, escrever o programa em linguagem Assembly RISC-V e executá-lo na placa Nexys A7.
- Em seguida, realizar uma simulação no Verilator com o mesmo programa. Pode adicionar os seguintes sinais: relógio do sistema, o registo do processador que armazena o valor a mostrar nos mostradores de 8 dígitos de 7 segmentos, e o temporizador regista RPTC_CNTR, RPTC_LRC, RPTC_HRC e RPTC_CTRL.
- Agora escreva o programa em C e execute-o na placa Nexys A7.
- Simular o seu programa C no Verilator, como na parte (b) para o programa Assembly RISC-V.

4. IMPLEMENTAÇÃO DE BAIXO-NÍVEL DO TEMPORIZADOR

Nesta secção, descrevemos primeiro a implementação de baixo-nível do módulo temporizador no Sistema RVfpga e depois propomos alguns exercícios onde primeiro modificará o módulo e depois o utilizará num programa para controlar os LEDs RGB disponíveis na placa Nexys A7.

A. Implementação de baixo-nível do temporizador

À semelhança do esquema que seguimos nos laboratórios anteriores, dividimos a análise do módulo temporizador em 2 fases.

- Integração do novo módulo no SweRVofX SoC (região sombreada à esquerda na Figura 1)
- Ligação entre o novo módulo e o Core SweRV EH1 (região sombreada à direita na Figura 1).

Note-se que, ao contrário dos laboratórios anteriores, este periférico (o temporizador) não está ligado fisicamente à placa Nexys A7. O temporizador é interno ao SweRVofX.

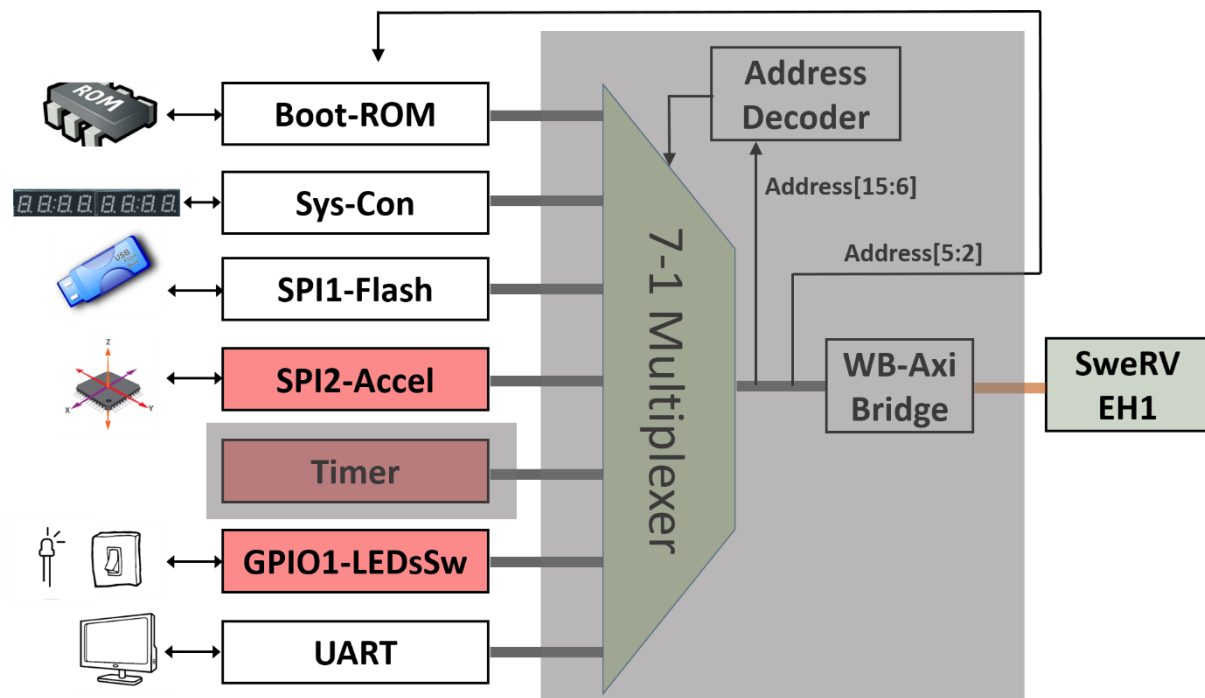


Figura 1. Análise do módulo temporizador em 2 fases

i. Integração do módulo temporizador no SoC

Nas linhas 361-379 do módulo **swervolf_core** (`[RVfpgaPath]/RVfpga/src/SweRVolfSoC/swervolf_core.v`) o módulo temporizador é instanciado (ver Figura 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 );

```

Figura 2. Integração do módulo temporizador (ficheiro **swervolf_core.v**).

Como de costume, a interface do módulo pode ser dividida em dois blocos: Sinais Wishbone (Tabela 3) e sinais externos de E/S (Tabela 4). Os sinais Wishbone permitem ao SweRV EH1 Core comunicar com o temporizador utilizando um modelo de

controlador/periférico. Os sinais de E/S externos, ligam o módulo temporizador com dispositivos externos; por exemplo, *pwm_pad_o* fornece o sinal de saída PWM ao operar no modo PWM descrito acima (terá de utilizar este sinal no Exercício 2 para ligar os módulos temporizadores com os LEDs RGB).

Tabela 3. Sinais Wishbone

Porto	Largura	Direção	Descrição
<i>wb_cyc_i</i>	1	Entradas	Indica um ciclo de barramento válido (seleção do núcleo)
<i>wb_adr_i</i>	15	Entradas	Entradas de endereços
<i>wb_dat_i</i>	32	Entradas	Entradas de dados
<i>wb_dat_o</i>	32	Saídas	Saídas de dados
<i>wb_sel_i</i>	4	Entradas	Indica bytes válidos no barramento de dados (durante o ciclo válido, este sinal deve ser 0xf)
<i>wb_ack_o</i>	1	Saída	Saída de confirmação (indica o término normal da transação)
<i>wb_err_o</i>	1	Saída	Saída de confirmação de erro (indica um cancelamento anormal da transação)
<i>wb_rty_o</i>	1	Saída	Não usado
<i>wb_we_i</i>	1	Entrada	Transação de escrita quando a nível alto
<i>wb_stb_i</i>	1	Entrada	Indica ciclo de transferência de dados válido
<i>wb_inta_o</i>	1	Saída	Saída de interrupção

Tabela 4. Sinais E/S Externos

Porto	Largura	Direção	Descrição
<i>gate_clk_pad_i</i>	1	Entrada	Relógio externo / Entrada de trinco
<i>capt_pad_i</i>	1	Entrada	Entrada de captura
<i>pwm_pad_o</i>	1	Saída	Saída de PWM
<i>oen_padoen_o</i>	1	Saída	Controlador de saída PWM habilitado (para tri-state ou de dreno aberto)

Como mostrado na linha 365 de Figura 2, os bits [5:2] do endereço fornecido pelo núcleo no sinal do barramento Wishbone (*wb_m2s_ptc_adr[5:2]*) são utilizados para selecionar um entre os 4 registos disponíveis (E/S em memória mapeada). Assim, podemos aceder ao registo RPTC_CNTR no endereço 0x80001200, registo RPTC_HRC no endereço 0x80001204, registo RPTC_LRC no endereço 0x80001208, e registo RPTC_CTRL no endereço 0x8000120C.

ii. Ligação entre o temporizador e o Core SweRV EH1

Como explicado nos laboratórios anteriores, os controladores de dispositivos estão ligados ao Core SweRV EH1 através de um Multiplexer (Figura 1). Lembre-se que o multiplexador 7:1 (Figura 3) é implementado no ficheiro

[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Interconnect/WishboneInterconnect/wb_intercon.v, que é instanciada 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** guardado aqui:

[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

Figura 3. 7-1 multiplexer que selecciona o periférico ligado ao CPU (ficheiro *wb_intercon.v*)

O multiplexer selecciona o periférico a ler ou escrever, ligando o CPU (*wb_io_** sinais - linhas 115-126 da Figura 3) com o barramento Wishbone de um periférico (linhas 127-138 da Figura 3), dependendo do endereço (linhas 110-111). Por exemplo, se o endereço gerado pelo CPU estiver no intervalo 0x80001200-0x8000123F, o módulo temporizador é selecionado, e assim os sinais *wb_io_** são ligados aos sinais *wb_ptc_**.

5. EXERCÍCIOS AVANÇADOS

Exercício 2. Modificar o RVfpgaNexys para ligar o sinal de saída PWM do temporizador (*pwm_pad_o*) a um dos dois LEDs RGB disponíveis na placa Nexys A7. Recomenda-se acrescentar esta nova capacidade ao sistema RVfpgaNexys atualizado que modificou nos Labs 6 e 7.

- A Digilent fornece as seguintes informações sobre os LEDs RGB disponíveis na placa Nexys A7: <https://reference.digilentinc.com/reference/programmable-logic/nexys-a7/reference-manual>
- Para resumir o documento acima, a placa contém dois LEDs RGB. Cada LED RGB tem três sinais de entrada que controlam os cátodos de três LEDs internos mais pequenos: um **vermelho (R)**, um **verde (G)**, e um **azul (B)**. Aplicando uma tensão num destes, o respectivo LED interno iluminará. O LED RGB emitirá uma cor dependente da combinação de LEDs internos que estão atualmente a ser iluminados. Por exemplo, fornecer uma tensão nos LEDs vermelho e azul emitirá uma cor púrpura. A Digilent recomenda fortemente a utilização da modulação por largura de pulso (PWM) ao controlar os LED RGB. Ativar qualquer uma das entradas para uma lógica estável '1' resultará na iluminação do LED a um nível desconfortavelmente brilhante. Isto pode ser evitado assegurando que nenhum dos sinais RGB seja acionado com mais de 50% de ciclo de funcionamento. Além disso, a utilização de PWM também aumenta consideravelmente a paleta de cores potenciais do LED tricolor. O ajuste individual do ciclo de funcionamento de cada cor entre 50% e 0% faz com que as diferentes cores sejam iluminadas a diferentes intensidades, permitindo que praticamente qualquer cor seja exibida.
- Crie três novos módulos temporizadores baseados no que já está incluído no SweRVolfX. Cada cor (vermelho, azul e verde) deve ser controlada por um módulo temporizador diferente, para que cada um possa receber uma tensão diferente.

- Utilize as seguintes gamas de endereços para mapear os registos para cada novo temporizador na memória:
 - i. Timer-2: 0x80001240-0x8000127F
 - ii. Timer-3: 0x80001280-0x800012BF
 - iii. Timer-4: 0x800012C0-0x800012FF
- Note que neste caso deve adicionar 3 novas entradas ao multiplexer que seleciona o periférico (Figura 1).
- Deve modificar o ficheiro de restrições tendo em conta que as 3 cores estão ligadas aos seguintes pinos de placa:
 - iv. LED16_B \leftrightarrow PIN R12
 - v. LED16_G \leftrightarrow PIN M16
 - vi. LED16_R \leftrightarrow PIN N15

Exercício 3. Implemente um programa que utilize o novo periférico para controlar o LED RGB, utilizando o valor fornecido pelos 16 interruptores. Utilizar os 5 interruptores mais à direita para ajustar o ciclo de funcionamento da cor azul, os 5 interruptores seguintes para ajustar o ciclo de funcionamento da cor verde, e os 5 interruptores seguintes para ajustar o ciclo de funcionamento da cor vermelha. (O interruptor mais à esquerda não será utilizado.)

- a. Primeiro, escreva o programa em Assembly do RISC-V.
- b. A seguir, escreva o programa em C.