

**THE IMAGINATION UNIVERSITY PROGRAMME**

**RVfpga Lab 8**

**Temporizadores**

# 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.

# 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.

1. **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 ptc\_ecgt* (‘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 (*ptc\_ecgt*). |
| 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*.

R

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).

# 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.

1. Primeiro, escrever o programa em linguagem Assembly RISC-V e executá-lo na placa Nexys A7.
2. 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\_CRTL.
3. Agora escreva o programa em C e execute-o na placa Nexys A7.
4. Simular o seu programa C no Verilator, como na parte (b) para o programa Assembly RISC-V.

# 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.

1. **Implementação de baixo-nível do temporizador**

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

1. Integração do novo módulo no SweRVolfX SoC (região sombreada à esquerda na Figura 1)
2. 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 SweRVolfX.



Figura 1. Análise do módulo temporizador em 2 fases

1. **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).

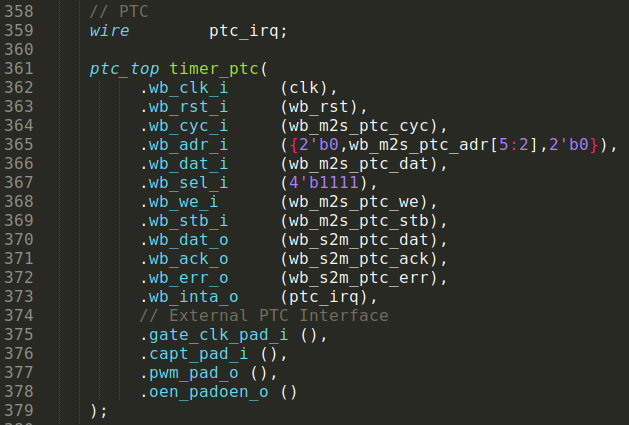


Figura 2. Integração do módulo temporizador (ficheiro *swervolf\_core.v*).

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** |
| wb\_cyc\_i | 1 | Entradas | Indica um ciclo de barramento válido (seleção do núcleo) |
| wb\_adr\_i | 15 | Entradas | Entradas de endereços |
| wb\_dat\_i | 32 | Entradas | Entradas de dados |
| wb\_dat\_o | 32 | Saídas | Saídas de dados |
| wb\_sel\_i | 4 | Entradas | Indica bytes válidos no barramento de dados (durante o ciclo válido, este sinal deve ser 0xf) |
| wb\_ack\_o | 1 | Saída | Saída de confirmação (indica o término normal da transação) |
| wb\_err\_o | 1 | Saída | Saída de confirmação de erro (indica um cancelamento anormal da transação) |
| wb\_rty\_o | 1 | Saída | Não usado |
| wb\_we\_i | 1 | Entrada | Transação de escrita quando a nível alto |
| wb\_stb\_i | 1 | Entrada | Indica ciclo de transferência de dados válido |
| wb\_inta\_o | 1 | Saída | Saída de interrupção |

Tabela 4. Sinais E/S Externos

|  |  |  |  |
| --- | --- | --- | --- |
| **Porto** | **Largura** | **Direção** | **Descrição** |
| gate\_clk\_pad\_i | 1 | Entrada | Relógio externo / Entrada de trinco |
| capt\_pad\_i | 1 | Entrada | Entrada de captura |
| pwm\_pad\_o | 1 | Saída | Saída de PWM |
| oen\_padoen\_o | 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.

1. **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*.



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

O multiplexer seleciona 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\_\**.

# 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:

# Timer-2: 0x80001240-0x8000127F

# Timer-3: 0x80001280-0x800012BF

# 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:
  + 1. LED16\_B 🡨🡪 PIN R12
    2. LED16\_G 🡨🡪 PIN M16
    3. LED16\_R 🡨🡪 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.)

1. Primeiro, escreva o programa em Assembly do RISC-V.
2. A seguir, escreva o programa em C.