PROGRAMA UNIVERSITARIO DE IMAGINATION

Práctica 8 RVfpga
Temporizadores

1. INTRODUCCIÓN

Los temporizadores hardware son periféricos comunes que se pueden encontrar en los microcontroladores y los SoCs. Se utilizan típicamente para generar una sincronización precisa. Los temporizadores incrementan o disminuyen un contador a una frecuencia fija, que a menudo es configurable, y luego interrumpen al procesador cuando el contador llega a cero o a un valor predefinido. Los temporizadores más sofisticados también pueden realizar otras funciones, como generar formas de onda de modulación de ancho de pulso (PWM por sus siglas en inglés) para controlar la velocidad de un motor o el brillo de una luz.

En esta práctica, utilizando una estructura similar a la de las prácticas anteriores, se describe primero la especificación de alto nivel del temporizador incluido en el Sistema RVfpga y luego se explica su implementación de bajo nivel. Se proponen también ejercicios tanto básicos como avanzados que muestran cómo usar y modificar un temporizador.

2. ESPECIFICACIÓN DE ALTO NIVEL DEL TEMPORIZADOR INCLUIDO EN EL SISTEMA RVfpga

En esta sección se analiza en primer lugar la especificación de alto nivel del temporizador utilizado en el Sistema RVfpga y a continuación se propone un ejercicio que utiliza este periférico.

A. Especificación de alto nivel del temporizador

El módulo temporizador utilizado en el Sistema RVfpga se ha obtenido de OpenCores (https://opencores.org/projects/ptc). Al descargar el paquete se proporciona un documento que describe la especificación de alto nivel del módulo (y que en nuestro caso se proporciona en: *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/ptc/docs/ptc_spec.pdf*). Aquí se resume el funcionamiento y características principales del módulo temporizador; sin embargo, la información completa se encuentra en el documento previamente mencionado.

El módulo temporizador tiene las siguientes características principales:

- Utiliza una interconexión Wishbone
- Servicio de contador/temporizador de 32 bits
- Ejecución simple o continua de PWM/Temporizador/Contador (PTC por sus siglas en inglés)
- Modo PWM (modulación de ancho de pulso) programable
- Funcionalidad de temporizador con el reloj del sistema y con fuentes externas de reloj
- Registros de Referencia y Captura HI/LO
- Control de tres estados para el controlador de salida PWM
- Las funcionalidades PTC pueden causar una interrupción en la CPU

La sección 4 del documento de especificaciones del módulo temporizador describe los registros de control y estado disponibles en éste, cada uno asignado a una dirección diferente (ver Tabla 1). La dirección base del temporizador en el Sistema RVfpga es **0x80001200**.

Tabla 1. Registros del temporizador

Nombre	Dirección	Ancho	Acceso	Descripción
RPTC_CNTR	0x80001200	1-32	R/W	Contador principal PTC
RPTC_HRC	0x80001204	1-32	R/W	Registro de referencia/captura PTC HI

RPTC_LRC	0x80001208	1-32	R/W	Registro de referencia/captura PTC LO
RPTC_CTRL	0x8000120C	9	R/W	Registro de control

El registro RPTC_CNTR es el registro contador real, y se incrementa en cada ciclo del reloj contador/temporizador. El registro RPTC_CTRL se utiliza para controlar el módulo temporizador; Tabla 2 muestra la función de cada uno de sus bits. Los registros RPTC_HRC y RPTC_LRC se utilizan como registros de referencia/captura.

Tabla 2. Bits del registro RPTC_CTRL

Bit	Acceso	Reset	Nombre y descripción
0	R/W	0	**EN** Cuando se pone a 1, el RPTC_CNTR se incrementa.
1	R/W	0	**ECLK** Selecciona la señal del reloj: reloj externo, a través de *ptc_ecgt* (1), o reloj del sistema (0).
2	R/W	0	**NEC** Se utiliza para seleccionar el flanco negativo/positivo y el período bajo/alto del reloj externo (*ptc_ecgt*).
3	R/W	0	**OE** Activa el controlador de salida PWM.
4	R/W	0	**SINGLE** Cuando se pone en 1, el RPTC_CNTR no se incrementa después de alcanzar un valor igual al valor de RPTC_LRC. Cuando se pone en 0, el RPTC_CNTR se reinicia después de que alcanza el valor del registro RPTC_LCR.
5	R/W	0	**INTE** Cuando está a 1, el PTC activa una interrupción cuando el valor de RPTC_CNTR es igual al valor de RPTC_LRC o RPTC_HRC. Cuando la señal está a 0, las interrupciones se enmascaran.
6	R/W	0	**INT** Este bit representa una interrupción pendiente. Cuando está en 1, hay una interrupción pendiente. Cuando en este bit se escribe un '1', la solicitud de interrupción se borra.
7	R/W	0	**CNTRRST** Cuando está a 1, se resetea el RPTC_CNTR. Cuando se pone a 0, el contador funciona normalmente.
8	R/W	0	**CAPTE** Cuando está a 1, el RPTC_CNTR se captura en los registros RPTC_LRC o RPTC_HRC. Cuando se pone en 0, la función de captura se enmascara.

TAREA: Localizar la declaración de los registros RPTC_CNTR, RPTC_HRC, RPTC_LRC y RPTC_CTRL en el módulo temporizador, así como la definición de sus direcciones (0x80001200, 0x80001204, 0x80001208 y 0x8000120C respectivamente). El módulo temporizador está disponible en la carpeta *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Peripherals/ptc*.

El temporizador puede funcionar de diferentes modos (a continuación se describen brevemente los modos que se utilizarán en esta práctica; véase la Sección 3 del documento de especificación del módulo temporizador para más detalles):

- **Modo temporizador/contador**: En este modo, el reloj del sistema o la referencia de reloj externo incrementan RPTC_CNTR si el contador está habilitado

(RPTC_CTRL[EN]=1). Cuando RPTC_CNTR es igual al RPTC_LRC, si RPTC_CTRL[INTE] está en 1, RPTC_CTRL[INT] se pone en 1.

- **Modo PWM**: Una señal de modulación de ancho de pulso (PWM) es un método para generar una señal analógica utilizando una fuente digital. Una señal PWM consiste en dos valores que definen su comportamiento: el *ciclo de trabajo (duty cycle)* y *la frecuencia*. El ciclo de trabajo describe la cantidad de tiempo que la señal está en alto como un porcentaje del tiempo total que tarda en completar un ciclo. La frecuencia se refiere a cuán a menudo se repite ese ciclo. Al hacer conmutar una señal digital entre apagado y encendido a una velocidad suficientemente rápida, y con un determinado ciclo de trabajo, la salida parecerá comportarse como una señal analógica de voltaje constante al proporcionar energía a otros dispositivos. Por ejemplo, una señal con un ciclo de trabajo del 50% (la mitad del tiempo de ciclo está a alto) y un voltaje a alto de 3,3 V parecería una carga analógica de 1,67 V (el voltaje promedio a lo largo del ciclo). La misma señal con un ciclo de trabajo del 33% parecería ser de 1,1 V. Para funcionar en modo PWM, se debe poner RPTC_CTRL[OE] a 1. Los registros RPTC_HRC y RPTC_LRC deben configurarse con el valor de los períodos alto y bajo de la señal de salida PWM: la señal PWM debe ir a nivel alto RPTC_HRC ciclos de reloj después del reset (del RPTC_CNTR); y la señal PWM debe ir a nivel bajo RPTC_LRC ciclos de reloj después del reset (del RPTC_CNTR).

3. EJERCICIO BÁSICO

Ejercicio 1. Escriba un programa que muestre una cuenta ascendente en el display de 7 segmentos de 8 dígitos. El valor debe cambiar aproximadamente una vez por segundo y, para crear este retardo, debe utilizar el módulo temporizador.

a. En primer lugar, escriba el programa en lenguaje ensamblador de RISC-V y ejecútelo en la placa Nexys A7.
b. Después, realice con el mismo programa una simulación en Verilator. Puede añadir las siguientes señales: el reloj del sistema, el registro del procesador que almacena el valor a mostrar en el display de 7 segmentos de 8 dígitos, y los registros del temporizador RPTC_CNTR, RPTC_LRC, RPTC_HRC y RPTC_CRTL.
c. Ahora escriba el programa en C y ejecútelo en la placa Nexys A7.
d. Simule su programa C en Verilator, como lo hizo en la parte (b) para el programa el lenguaje ensamblador de RISC-V.

4. IMPLEMENTACIÓN DE BAJO NIVEL DEL TEMPORIZADOR

En esta sección se describe primero la implementación de bajo nivel del módulo temporizador en el Sistema RVfpga y a continuación se proponen algunos ejercicios en los que el usuario primero modificará el módulo y después lo utilizará en un programa para controlar los LED tricolor disponibles en la placa Nexys A7.

A. Implementación de bajo nivel del temporizador

De forma similar al esquema que se ha seguido en las prácticas anteriores, se divide en fases el análisis del módulo temporizador.

1. Integración del nuevo módulo en SweRVolfX (región sombreada a la izquierda en la Figura 1).

2. Conexión entre el nuevo módulo y el core SweRV EH1 (región sombreada a la derecha en la Figura 1).

Tenga en cuenta que, a diferencia de las prácticas anteriores, este periférico (el temporizador) no está conectado físicamente a la placa Nexys A7. El temporizador es interno a SweRVolfX.

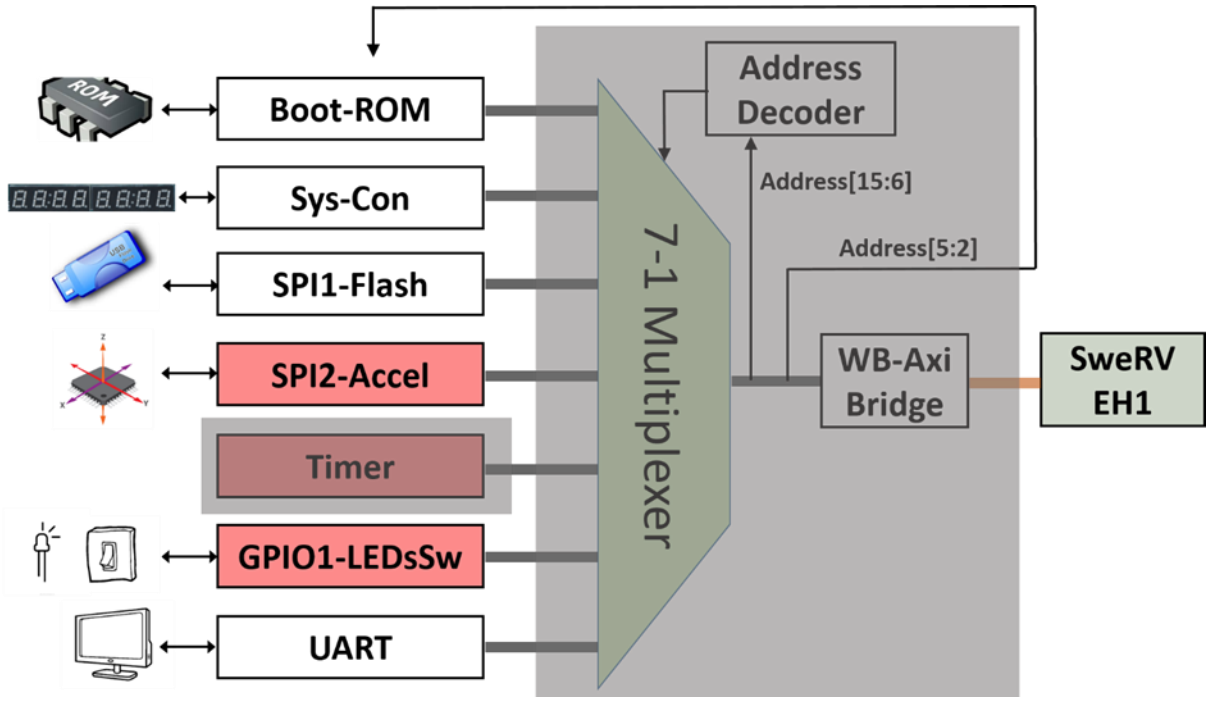

Figura 1. Análisis del módulo temporizador en 2 fases

i. Integración del módulo del temporizador en el SoC

En las líneas 361-379 del módulo **swervolf_core** (*[RVfpgaPath]/RVfpga/src/SweRVolfSoC/swervolf_core.v*) se instancia el módulo temporizador (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. Integración del módulo temporizador (archivo *swervolf_core.v*).

Como de costumbre, la interfaz del módulo puede dividirse en dos bloques: señales Wishbone (Tabla 3) y señales de Entrada/Salida externas (Tabla 4). Las señales Wishbone permiten al core SweRV EH1 comunicarse con el temporizador mediante un modelo controlador/periférico. Las señales de Entrada/Salida externas, conectan el módulo temporizador con dispositivos externos; por ejemplo, *pwm_pad_o* proporciona la señal de salida PWM cuando se opera en el modo PWM descrito anteriormente (tendrá que utilizar esta señal en el Ejercicio 2 para conectar los módulos temporizador con los LED tricolor).

Tabla 3. Señales Wishbone

Puerto	Ancho	Dirección	Descripción
wb_cyc_i	1	Entradas	Indica un ciclo de bus válido (selección de core)
wb_adr_i	15	Entradas	Entradas de direcciones
wb_dat_i	32	Entradas	Entradas de datos
wb_dat_o	32	Salidas	Salidas de datos
wb_sel_i	4	Entradas	Indica los bytes válidos en el bus de datos (durante el ciclo válido debe ser 0xf)
wb_ack_o	1	Salida	Salida de *acknowledgment* (indica la terminación normal de la transacción)
wb_err_o	1	Salida	Salida *acknowledgment* de error (indica una terminación anormal de la transacción)
wb_rty_o	1	Salida	No se usa
wb_we_i	1	Entrada	Transacción de escritura cuando su valor es 1
wb_stb_i	1	Entrada	Indica un ciclo de transferencia de datos válido
wb_inta_o	1	Salida	Salida de interrupción

Tabla 4. Señales de E/S externas

Puerto	Ancho	Dirección	Descripción

gate_clk_pad_i	1	Entrada	Reloj externo / Entrada de puerta
capt_pad_i	1	Entrada	Entrada de captura
pwm_pad_o	1	Salida	Salida PWM
oen_padoen_o	1	Salida	Habilitación del controlador de salida PWM (para controlador de tres estados o de drenador abierto)

Como se muestra en la línea 365 de la Figura 2, los bits [5:2] de la dirección proporcionada por el core en la señal del bus Wishbone (*wb_m2s_ptc_adr[5:2]*) se utilizan para seleccionar uno de los 4 registros disponibles (Entrada/Salida mapeada en memoria). Así, se puede acceder al registro RPTC_CNTR en la dirección 0x80001200, al registro RPTC_HRC en la dirección 0x80001204, al registro RPTC_LRC en la dirección 0x80001208 y al registro RPTC_CTRL en la dirección 0x8000120C.

ii. Conexión entre el temporizador y el core SweRV EH1

Como se ha explicado en prácticas anteriores, los controladores de dispositivo están conectados con el core SweRV EH1 a través de un multiplexor (Figura 1). Recuerde que el multiplexor 7:1 (Figura 3) se implementa en el fichero *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Interconnect/WishboneInterconnect/wb_intercon.v*, el cual se instancia en las líneas 104-205 del fichero *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/Interconnect/WishboneInterconnect/wb_intercon.vh* . Este último archivo está incluido en la línea 168 del módulo **swervolf_core** que se encuentra aquí: *[RVfpgaPath]/RVfpga/src/SweRVolfSoC/swervolf_core.v*.

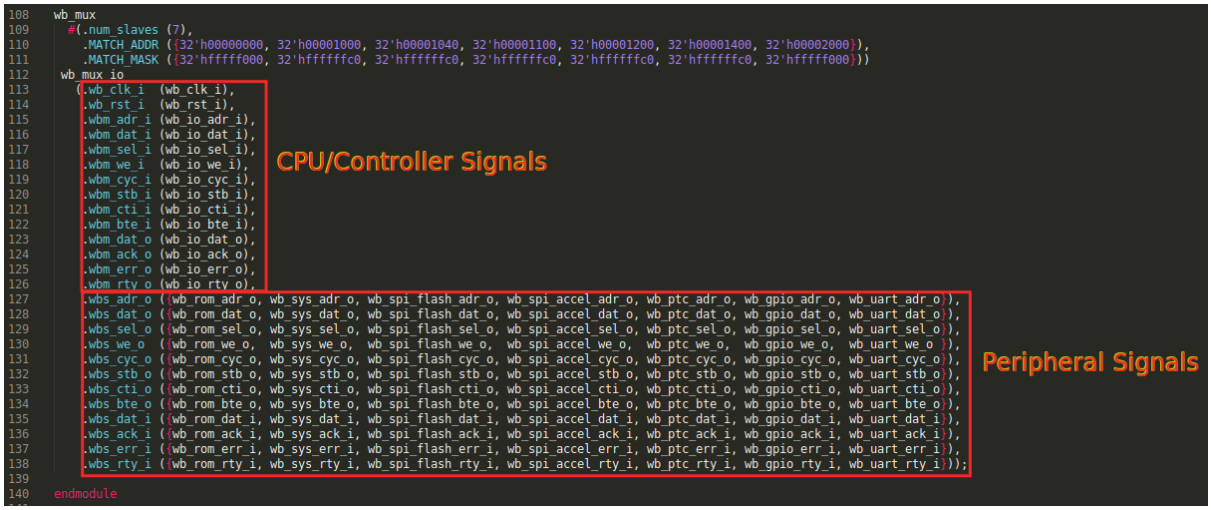

Figura 3. 7-1 Multiplexor que selecciona el periférico conectado con la CPU (archivo *wb_intercon.v*)

El multiplexor selecciona qué periférico leer o escribir, conectando la CPU (señales *wb_io_** - líneas 115-126 de la Figura 3) con el Bus Wishbone de un periférico (líneas 127-138 de la Figura 3), dependiendo de la dirección (líneas 110-111). Por ejemplo, si la dirección generada por la CPU está en el rango 0x80001200-0x8000123F, se selecciona el módulo temporizador, y así las señales *wb_io_** se conectan con las señales *wb_ptc_**.

5. EJERCICIOS AVANZADOS

Ejercicio 2. Modifique RVfpgaNexys para conectar la señal de salida PWM del temporizador (*pwm_pad_o*) a uno de los dos LEDs tricolor disponibles en la placa Nexys A7. Se recomienda añadir esta nueva funcionalidad al sistema RVfpgaNexys actualizado que el usuario modificó en las Prácticas 6 y 7.

- Digilent proporciona la siguiente información sobre los LED tricolor disponibles en la placa Nexys A7: https://reference.digilentinc.com/reference/programmable-logic/nexys-a7/reference-manual
- Para resumir el documento anterior, la placa contiene dos LED tricolor. Cada LED tricolor tiene tres señales de entrada que controlan los cátodos de tres LEDs internos más pequeños: uno **rojo**, uno **azul** y uno **verde**. Llevando uno de estos cátodos a nivel alto se iluminará el LED interno correspondiente. El LED tricolor emitirá un color que dependerá de la combinación de los LEDs internos que se estén iluminando en ese momento. Por ejemplo, si se llevan a alto el rojo y el azul, se emitirá un color púrpura. Digilent recomienda vivamente el uso de la modulación de ancho de pulso (PWM) al controlar los LED tricolores. Si se lleva cualquiera de las entradas a un "1" lógico estable, el LED se iluminará con un nivel de brillo incómodo. Puede evitar esto asegurándose de que ninguna de las señales tricolor se active con un ciclo de trabajo de más del 50%. Además, el uso de PWM también amplía enormemente la paleta de colores potenciales del LED tricolor. Ajustando individualmente el ciclo de trabajo de cada color entre el 50% y el 0% hace que los diferentes colores se iluminen a diferentes intensidades, permitiendo que se muestre prácticamente cualquier color.
- Crear tres nuevos módulos temporizador basados en el ya incluido en SweRVolfX. Cada color (rojo, azul y verde) debe ser gestionado por un módulo temporizador diferente, de modo que cada uno pueda recibir un voltaje diferente.
- Utilice los siguientes rangos de direcciones para asignar los registros de cada nuevo temporizador a la memoria:
    i. Temporizador-2: 0x80001240-0x8000127F
    ii. Temporizador-3: 0x80001280-0x800012BF
    iii. Temporizador-4: 0x800012C0-0x800012FF

  Tenga en cuenta que en este caso debe añadir 3 nuevas entradas al multiplexor que selecciona el periférico (Figura 1).
- Debe modificar el archivo de restricciones teniendo en cuenta que los 3 colores se conectan a los siguientes pines de la placa:
    iv. LED16_B ←→ PIN R12
    v. LED16_G ←→ PIN M16
    vi. LED16_R ←→ PIN N15

Ejercicio 3. Implemente un programa que utilice el nuevo periférico para controlar el LED tricolor, utilizando el valor proporcionado por los 16 interruptores. Utilice los 5 interruptores de la derecha para ajustar el ciclo de trabajo del color azul, los 5 interruptores siguientes para ajustar el ciclo de trabajo del color verde, y los 5 interruptores siguientes para ajustar el ciclo de trabajo del color rojo. (El interruptor más a la izquierda no se usará).

a. Primero, escriba el programa en ensamblador de RISC-V.
b. A continuación, escriba el programa en C.