Creación de un driver de protocolo

Práctica 1, Sistemas Lógicos Programables Avanzados.

Hipólito Guzmán Miranda
Departamento de Ingeniería Electrónica
Universidad de Sevilla
hguzman@us.es

Creación de un driver de protocolo

Se desea crear una entidad VHDL, no necesariamente sintetizable, para facilitar los esfuerzos de verificación de un diseño digital. Esta entidad recibe una palabra de 32 bits de datos junto con un campo que indica que el dato es válido.

Se proporciona un `package` VHDL en la página de la asignatura que describe un `record` que incluye dichos datos. Se recuerda que, para poder utilizarlo, además de añadirlo al proyecto, se debe incluir utilizando la sentencia `use work.protocol_common.all;` en la sección `LIBRARY` de los ficheros VHDL que necesiten hacer uso del mismo. Se permite añadir código al package si se estima necesario o conveniente.

Funcionamiento del driver:

El driver debe recibir como entradas una señal de reloj, `clk`, y una entrada `input_tran`, del tipo `record` mencionado anteriormente. Como salidas debe tener las señales del protocolo: `data`, `ena`, `startp` y `endp`. Se describen los puertos de la entidad en la siguiente tabla:

Nombre	Tipo de dato	Dirección	Función
`input_tran`	`protocol_type`	`in`	Transacción de entrada
`clk`	`std_ulogic`	`in`	Reloj activo por flanco de subida. Los cambios en las salidas serán síncronos con este reloj.
`data`	`std_ulogic_vector` (de anchura dependiente del DNI del alumno)	`out`	Datos de salida
`ena`	`std_ulogic`	`out`	Señal de habilitación del protocolo
`startp`	`std_ulogic`	`out`	Señal de inicio de transmisión
`endp`	`std_ulogic`	`out`	Señal de fin de transmisión

Cuando el campo `valid` de `input_tran` valga '1', el driver debe enviar el dato que haya en ese momento en el campo `data` de `input_tran` según el protocolo que se describe en el siguiente apartado. Cuando se desee enviar una transacción, el módulo que utilice el driver pondrá `input_tran.valid` a '1' durante un ciclo de reloj.

Descripción del protocolo:

El protocolo consiste en el envío de una palabra de 32 bits de datos, serializada en bloques de entre 1 y 16 bits que se envían secuencialmente (es decir, se envían los datos bloque a bloque).

Cuando se deseen enviar datos, debe activarse la señal `ena`. Tras ésto, debe habilitarse durante 1 ciclo de reloj la señal `startp`, tras lo cual deben enviarse los datos en el orden correcto por el puerto `data`. Cuando no se estén enviando datos, la salida `data` debe ponerse en alta impedancia (`'Z'`). Finalmente, debe habilitarse durante un ciclo de reloj la señal `endp` y deshabilitarse la señal `ena`. Entre cualquiera de estos pasos puede haber ciclos de espera según se indica en el siguiente apartado.

Personalización con DNI / NIE:

Utilizando los dígitos del DNI / NIE del alumno:

d7 / letra1	d6	d5	d4	d3	d2	d1	d0	letra0

Se calcularán los siguientes parámetros del protocolo:

Anchura de los datos serializados (anchura de los bloques de datos):
d0 módulo 5:

- 0: anchura de 16 bits
- 1: anchura de 8 bits
- 2: anchura de 4 bits
- 3: anchura de 2 bits
- 4: anchura de 1 bit

Orden de los datos serializados:
d1 es:

- Par: los bloque de datos se envían empezando por los más significativos y terminando en los menos significativos
- Impar: los bloques de datos se envían empezando por los menos significativos y terminando en los más significativos

Polaridad de la señal `ena`:
d2 es:

- Par: `ena` es activa a nivel bajo
- Impar: `ena` activa a nivel alto

Polaridades de las señales `startp` y `endp`:
d3 es:

- Par: activas a nivel bajo
- Impar: activas a nivel alto

Ciclos, tras la activación de ena, que hay que esperar antes de activar la señal startp:
d3

Ciclos, tras la activación de startp, que hay que esperar antes de asignar el valor del primer bloque de datos a `data`:
d4

Duración, en ciclos, de cada bloque de datos:
d5 * 10 + d4

Si este resultado sale 0 se tomará como valor un 1. El valor en `data` debe mantenerse constante durante este número de ciclos.

Ciclos, a contar desde el fin del último ciclo del último bloque de datos, que hay que esperar antes de activar la señal `endp`:
d6

Ciclos, tras la activación de la señal `endp`, que hay que esperar antes de deshabilitar `ena`:
d6 + d5

Ciclos mínimos, tras la deshabilitación de `ena`, que hay que esperar antes de activar de nuevo `ena` si se quiere enviar una nueva transacción:
d5 + d4

Realización y evaluación de la práctica:

Se debe desarrollar el driver de protocolo adaptado al DNI del alumno (DNI/NIE/etc). Se recomienda dibujar los diagramas de forma de onda del protocolo resultante antes de describir el VHDL.

Para el desarrollo del driver, una opción es utilizar procesos sin lista de sensibilidad que hagan uso de la sentencia `wait`. La sentencia `wait` puede utilizarse de las siguientes formas posibles:

```
wait for <time>;  -- Espera el tiempo indicado, por ejemplo: wait for 10 ns;

wait on <sensitivity list>  -- Espera a que cambie alguna de las señales indicadas en la lista de sensibilidad, por ejemplo: wait on enable, disable;

wait until <condition>  -- Espera a que cambie alguna de las señales que aparecen en la condición y se cumpla dicha condición, por ejemplo wait until enable = '1';

wait;  -- Sin más argumentos, espera para siempre en esta sentencia, evitando que se llegue al final del process.
```

Se valorarán ampliaciones sobre la funcionalidad básica tales como:

- Informar de cuándo se inicia y se termina una transacción sobre el bus utilizando la sentencia `report`.
- Considerar qué ocurre si el driver recibe una nueva transacción mientras está ocupado e implementar una solución.

Se debe desarrollar una memoria de la práctica en la que se describa el trabajo realizado y se demuestre que se ha implementado correctamente el driver de protocolo. La memoria debe incluir los cálculos de los parámetros del protocolo en función del DNI del alumno y la forma de onda esperada del mismo. Para esto, https://wavedrom.com/editor.html les puede resultar un recurso útil para generar diagramas de onda. La demostración de funcionamiento debe basarse en simulaciones y capturas de las formas de onda resultantes de las mismas. Adicionalmente a la memoria de la práctica, debe entregarse todo el código desarrollado, incluyendo los testbenches y el package `protocol_common`.