Práctica 1. Diseño de Sistemas Secuenciales.

Antonio María Franqués García (77791084R), grupo A2A.

Tarea 1

En esta tarea se implementa un contador ascendente-descendente binario de 8 bits; por lo que podrá contar de 0 a 255. Dispone de un *reset* asíncrono que permitirá poner de inmediato a cero el estado del contador, independientemente del valor del *reloj* en ese momento. Siempre que la señal de habilitación esté activa (a nivel bajo) permitirá cargar en paralelo (de golpe) una secuencia de 8 bits, a partir de la cual se empezará a contar. Si la señal de habilitación no está activa no se atenderá dicha entrada.

Dificultades encontradas durante la realización: mínimas, aunque se puede destacar que he intentado compactar el código para una más fácil lectura.

Comentarios sobre la simulación (*simt1.vwf*): en esta tarea he desfasado ligeramente la señal de *reloj* para que el flanco de bajada nunca coincida con los cambios de las demás señales.
Empiezo activando el *reset* para así hacer la puesta a punto. Al cabo de unos pocos instantes el *reset* se deshabilita y empieza la cuenta desde cero aún y cuando hay una serie de bits preparados para ser cargados. Tras otros pocos instantes se cargan dichos bits y la cuenta empieza desde estos, aunque sin dejarles mucho tiempo a más son reemplazados por otros que finalmente hemos decidido que sean quienes empiecen la cuenta (en medio de una carga y otra hay algunos bits espurios que gracias a que no tenemos el bit de carga activado serán ignorados). En un momento dado después de esta última carga se observa que la cuenta llega a su fin y vuelve a empezar desde cero, tanto si la cuenta es ascendente como descendente. Otra habilitación del *reset* hará que la cuenta se reinicie, para que unos momentos más tarde sean cargados otros bits, éstos seguirán contando ascendente y descendentemente a lo largo de otro tiempo. Para una más cómoda visualización de los resultados de la simulación expreso tanto *load* como *count* en decimal (botón derecho sobre nombre señal/Radix/Unsigned Decimal).

Tarea 2

En esta tarea se implementa un contador ascendente-descendente binario de módulo parametrizable; por lo que el usuario tan sólo deberá modificar el parámetro correspondiente al número de estados deseado. Dispone de un *reset* síncrono que prevalecerá sobre el resto de señales. El contador tan sólo modificará su estado en el caso de que la señal de *enable* esté activa. Cada vez que el contador llegue al fin de cuenta se activará una señal de control.

Dificultades encontradas durante la realización:

- Hallar el número de bits necesarios a partir del módulo del contador; la función CLogB2 ha solucionado el problema.
- Adaptar para cualquier número de bits el parámetro *fincuenta* que se utiliza para comparar si el estado actual es el último; *fincuenta={WIDTH{1'b1}}* donde *WIDTH* indica el número de bits que quiero "poner a uno" ha solucionado el problema.
- Darme cuenta de que la señal de control de fin de cuenta tan sólo debe activarse en el estado correspondiente al fin de cuenta, desactivándose hasta volver a llegar a este.

Comentarios sobre la simulación (*simt2.vwf*): Empiezo activando el *reset* para así hacer la puesta a punto. Desde el inicio está preparado el bit de cuenta ascendente, también la señal habilitadora está activa para que nada más se desactive el *reset* empiece la cuenta. Unos instantes después se hace una cuenta descendente que es interrumpida por la señal de *reset*, por lo que en cuanto éste se desactive se habilitará la señal de control conforme se ha alcanzado el

fin de cuenta (en este caso regresivamente). Más tarde se desactivará el habilitador de contador, para que cuando vuelva a activarse, la cuenta pase a ser ascendente (el bit *count_up* está activo); tras unos instantes se alcanza el fin de cuenta y por ende se activa nuevamente la señal de control. Finalmente la cuenta ascendente sigue ininterrumpidamente.

Tarea 3

En esta tarea se implementa un registro de desplazamiento de 7 bits. Eso es, si introducimos un nivel alto por la entrada serie deberemos esperar 7 tiempos de reloj para tenerlo en la salida serie (siempre y cuando el habilitador de desplazamiento permanezca activo y el *reset* deshabilitado). Como ya he mencionado, tan sólo se habilitará el desplazamiento en caso de que el bit *enable* se encuentre a nivel alto y el bit de *reset* (síncrono) a nivel bajo. En caso de que el *reset* esté activo el estado del registro pasará a valer cero, independientemente de que esté habilitado el desplazamiento; es por eso que decimos que el *reset* es de máxima prioridad. Dispone de una salida paralelo para observar de golpe el estado de los 7 bits y también de una salida serie expresa para observar el estado del último.

Dificultades encontradas durante la realización:

- Hacer las dos asignaciones no bloqueantes: mover el valor de los 6 primeros bits a los últimos 6 bits y mover el valor de la entrada serie al primer bit.
  out[5:0] <= out[6:1];
  out[6] <= s_in;
- Darme cuenta de que la salida serie es lo mismo que el bit de menor peso de la salida paralelo una vez fuera del *always*.

Comentarios sobre la simulación (*simt3.vwf*): Empiezo activando el *reset* para así hacer la puesta a punto. Aunque el habilitador de desplazamiento esté activo desde el principio tan sólo se permitirá la entrada serie y el posterior desplazamiento una vez el *reset* pase a deshabilitarse. El primer bit introducido a nivel alto llega ininterrumpidamente a la salida serie, sin embargo los dos que he introducido mientras llegaba el primero no alcanzan todo el recorrido por culpa de que la señal de *reset* se ha habilitado durante unos instantes. La señal habilitadora de desplazamiento restará activa inútilmente unos instantes más ya que ningún bit más es introducido incluso después de la desactivación del *reset*; se están desplazando ceros. Más tarde una serie de bits espurios están en la entrada serie durante el tiempo que no nos interesa desplazar (*enable* deshabilitado) por lo que son ignorados. Posteriormente se habilita el desplazamiento, se desplazan dos bits contiguos que llegan satisfactoriamente, hay un instante donde se deshabilita el desplazamiento ya que no hay bits a desplazar, al siguiente se habilita nuevamente para que un nuevo bit pueda completar los 7 tiempos de recorrido, al llegar este se introducen otros tres contiguos que son pausados unos instantes antes de completar finalmente su recorrido; inmediatamente después de recibir el último se deshabilita el desplazamiento.

Tarea 4

En esta tarea se implementa un juego de luces parecido al del famoso coche fantástico "kit". Para ello se utilizan los 8 *led* verdes de los que dispone la FPGA y un interruptor habilitador.

Dificultades encontradas durante la realización:

- Instanciar correctamente los módulos del contador y del registro realizados en las anteriores tareas.
- Enlazar correctamente las entradas y salidas de esas instancias mediante el uso de *wire*.
- Darme cuenta de que podía usar el *wire* asignado a la salida del registro para evaluar la sentencia *case* que asigna a cada *estado Johnson* un estado determinado de los *led*.

Comentarios sobre la simulación: La simulación de esta tarea se ha probado directamente en la *FPGA*. Tras observar que el juego de luces correspondía al esperado se ha considerado que funciona correctamente.