Manual Jetpack

Tecnología y Organización de Computadores

PiKey Team- P_KT :

Jesús Aguirre Pemán
Enrique Ballesteros Horcajo
Mayra Alexandra Castrosqui Florián
Jaime Dan Porras Rhee
Ignacio Iker Prado Rujas

7 de Marzo de 2014

Índice general

1. Ficheros y dependencias

El módulo central de nuestro proyecto es `choca.vhd`. Es ahí donde se gestionan las posiciones de Barry, el fondo, obstáculos, monedas... Sin embargo, para cargar las imágenes del juego desde memoria utilizamos varias ROMs:

- `rom_rgb_9b_barryair.vhd`: Memoria ROM que contiene una de las dos imágenes de Barry. Hay dos para generar el efecto de que el muñeco camina sobre el suelo.
- `rom_rgb_9b_barryair2-5.vhd`: Memoria ROM que contiene la otra imagen de Barry.
- `rom_rgb_9b_lab.vhd`: Memoria ROM que contiene los datos del fondo de los dos primeros niveles del juego.
- `rom_rgb_9b_mapa_facil.vhd`: Memoria ROM que contiene los obstáculos del primer nivel del juego.
- `rom_rgb_9b_flappy-nivelBW.vhd`: Memoria ROM que contiene los datos de los obstáculos del segundo nivel del juego.
- `rom_rgb_9b_arboles.vhd`: Memoria ROM que contiene los datos de los árboles del fondo del tercer nivel del juego.
- `rom_rgb_9b_nubes.vhd`: Memoria ROM que contiene el fondo con las nubes y el cielo del tercer nivel del juego.
- `rom_rgb_9b_nivelfuegoBW.vhd`: Memoria ROM que contiene los obstáculos del tercer nivel del juego.
- `rom_rgb_9b_game-over-negro.vhd`: Memoria ROM que contiene los datos de la imagen del "Game Over".

Además, utilizamos distintas frecuencias para Barry, el fondo y los obstáculos, pues cada uno de ellos se mueve a una velocidad diferente. Para ello tenemos varios ficheros divisores:

- `divisor.vhd`: Tiene como entrada al *clk_100M* (con frecuencia 100 MHz) y salida *clk_1*, cuya frecuencia es de 12,5 MHz. Éste, a su vez, va conectado directamente al *clk*. Por tanto, divide por 8.
- `divisor_choques`: Tiene como entrada al *clk_100M* (con frecuencia 100 MHz) y salida *relojChoques*, cuya frecuencia es de 1 MHz. Por tanto, divide por 100.
- `divisor_munyeco.vhd`: Tiene como entrada al *clk_100M* (con frecuencia 100 MHz) y salida *relojMunyeco*, cuya frecuencia es de 190,7 Hz. Por tanto, divide por 524288.
- `divisor_corre.vhd`: Tiene como entrada al *clk_100M* (con frecuencia 100 MHz) y salida *pasa_tiempo*, cuya frecuencia es de 5,96 Hz. Por tanto, divide por 16777218.
- `divisor_movimiento_fondo.vhd`: Tiene como entrada al *clk_100M* (con frecuencia 100 MHz) y salida *relojMovFondo*, cuya frecuencia es de 95,4 Hz. Por tanto, divide por 1048576.
- `divisor_movimiento_inter_fondo.vhd`: Para el reloj del fondo intermedio del tercer escenario. Tiene como entrada al *clk_100M* (con frecuencia 100 MHz) y salida *relojMovFondoInter*, cuya frecuencia es de 127,3 Hz. Por tanto, divide por 785410.
- `divisor_movimiento_obstaculos.vhd`: Tiene como entrada al *clk_100M* (con frecuencia 100 MHz) y salida *relojMovimiento*, cuya frecuencia es de 152,5 Hz. Por tanto, divide por 655360.

- `divisor_movimiento_moneda.vhd`: Tiene como entrada al *clk_100M* (con frecuencia 100 MHz) y salida *relojMovMoneda*, cuya frecuencia es de 124 Hz. Por tanto, divide por 806432.

Con el fichero `contador.vhd` controlaremos el número de monedas cogidas, que se mostrarán en los displays 7 segmentos de la placa mediante el fichero `pines.ucf`.

Mediante el fichero `control_teclado.vhd` controlamos las teclas que pulsamos: la barra espaciadora para hacer ascender a Barry, y la 'P' para pausar la partida. Es importante tener en cuenta que si mientras estamos pulsando la barra espaciadora además pulsamos cualquier otra tecla (distinta de 'P') el controlador deja de detectar el espacio debido a la nueva tecla pulsada y Barry caerá hacia el suelo.

El `randomGenerator.vhd` es el encargado de generar coordenadas y aleatorias para la posición inicial de la moneda en pantalla. La coordenada x es fija pues siempre aparecen desde el fondo de la pantalla (el lado opuesto de donde se sitúa Barry).

El `gestorCambioNivel.vhd`

Como ya se ha comentado, todos estos ficheros son componentes de `choca.vhd`, y están conectados a él con las instrucciones correspondientes.

2. nombre.bit

`vgacore.bit`. O qué??

3. Pines y significado

Conectaremos el teclado con los pines:

```
NET PS2CLK LOC = B16;
NET PS2DATA LOC = E13;
NET PS2CLK CLOCK_DEDICATED_ROUTE = FALSE;
```

Mostraremos el número de monedas mediante los displays siete segmentos de la placa extendida:

Placa izquierda:

```
NET displayizq<0> LOC = H14;
NET displayizq<1> LOC = M4;
NET displayizq<2> LOC = P1;
NET displayizq<3> LOC = N3;
NET displayizq<4> LOC = M15;
NET displayizq<5> LOC = H13;
NET displayizq<6> LOC = G16;
```

Placa derecha:

```
NET displaydcha<0> LOC = E2;
NET displaydcha<1> LOC = E1;
NET displaydcha<2> LOC = F3;
NET displaydcha<3> LOC = F2;
NET displaydcha<4> LOC = G4;
NET displaydcha<5> LOC = G3;
NET displaydcha<6> LOC = G1;
```

Por último, conectaremos los colores y la posición en la pantalla:

```
NET rgb<0> LOC = C9; #BLU0;
```

```
NET rgb<1> LOC = E7; #BLU1;
NET rgb<2> LOC = D5; #BLU2;
NET rgb<3> LOC = A8; #GRN0;
NET rgb<4> LOC = A5; #GRN1;
NET rgb<5> LOC = C3; #GRN2;
NET rgb<6> LOC = C8; #RED0;
NET rgb<7> LOC = D6; #RED1;
NET rgb<8> LOC = B1; #RED2;
NET hsyncb LOC = B7; #HSYNC#
NET vsyncb LOC = D8; #VSYNC#
```

4. Otros aquelarres de inicialización

Nada en especial que comentar aquí. Lo único reseñable es que dependiendo del puesto de laboratorio en el que se ejecute puede haber diferencias en la posición de la zona de juego o en los colores (en algún caso salían los colores rosados). Ésto es debido a la configuración y uso de cada pantalla concreta.

5. Cómo se juega

El juego comienza en el primer nivel con 0 monedas recogidas. Después, tras el despegue, simplemente se debe pulsar la barra espaciadora para ascender y soltarla para descender. El objetivo es esquivar todos los obstáculos, y recoger el mayor número de monedas posible.

A medida que se recogen monedas, el juego cambiará de nivel. Si desea pausar el juego se debe pulsar la letra 'P'. Para salir de la pausa simplemente volver a pausar la barra espaciadora. El juego acaba al chocar con algún obstáculo. [1]

6. Qué espero ver

Se trata de un juego para FPGA inspirado en el Jetpack Joyride del grupo Halfbrick Studios, que está disponible de manera gratuita para plataformas Android e iOS. El juego consta de 3 niveles diferentes:

- **Nivel 1**: obstáculos espaciales y fondo de laboratorio.
- **Nivel 2**: tuberías estilo *Flappy Bird* como obstáculos y fondo de laboratorio.
- **Nivel 3**: obstáculos de fuego y fondo inspirado en *Super Mario*, con árboles y nubes en dos capas.

Cuando Barry está en el suelo, se alternarán dos fotogramas suyos (que están en sus ROMs correspondientes, ver el primer apartado de este documento) dando la sensación de que camina.

El número de monedas recogidas se mostrará por los displays. Al resetear volverá la cuenta a 0. Empezamos en el nivel 1, y al conseguir 5 monedas pasaremos al nivel 2. Cuando obtengamos 5 monedas más, avanzaremos al nivel 3, y recogiendo otras 5 monedas volveremos al nivel 1. Sucesivamente, cada 5 monedas recogidas cambiaremos de nivel hasta llegar a completar el desafío: 60 monedas. Si se alcanza dicho número el juego permanece en el nivel 3.

[1] En el video de presentación se puede observar que después de haber chocado con un obstáculo, al pulsar el espacio se puede seguir jugando normalmente. En este caso está puesto así a propósito ya que de este modo es más fácil grabar el video de la presentación de cada nivel y testear el funcionamiento.

Al pausar el juego, si hubiera alguna moneda en pantalla se pausará también, reanudando su camino al continuar jugando. No pasa ésto con el "Game Over", ya que al chocar con un obstáculo, las monedas seguirán su curso, como es lógico. [2]

[2] Ésto se puede apreciar en el video en el escenario 2.