Ejercicio 2. Diseño Hardware DE2-115. Cuestiones Prácticas Co-diseño

Rocabado Rocha, Jose Luis:

Responder las preguntas utilizando el espacio necesario. Formatear al gusto.

Ejercicio 2: Diseño de un hardware de referencia para DE2-115

Ejercicio 0: Proyecto software test_mtl

1. ¿Cómo se puede modificar la velocidad de movimiento del gusanito por el display 7-segmentos? ¿Se podría establecer una velocidad exacta de cambio cada 50 ms? ¿Por qué?
   Se puede modificar la velocidad de movimiento del gusanito cambiando la variable *"delay_count"* el código del documento "test_mtl.c" que actúa como contador añadiendo un retardo hasta que la cuenta termine.
   El problema es que el tiempo de retraso depende de la frecuencia de ejecución de instrucciones en ensamblador por lo que es difícil establecer una velocidad exacta de 50ms. La forma de obtener un retraso exacto sería utilizando un contador hardware que trabaje a la frecuencia de reloj del sistema de 50MHz cuyo módulo de cuenta sea de 2500000.
2. ¿Qué función tiene la línea de código: `while (*KEY_ptr);`? ¿Qué ocurre si la eliminamos?
   Su función es detener la ejecución hasta que se suelte el pulsador. Si no estuviera dentro del código los patrones de los siete segmentos cambiarían constantemente.
3. En la subrutina de dibujar un cuadrado en la pantalla `MTL_box`, aparece la siguiente línea de código: `offset = (row << 9) + col;` ¿Que función realiza esta línea de código? ¿A que se debe ese 9? ¿Se puede modificar el 9 por otro valor? ¿Cuál sería el efecto?
   Esta línea de código nos permite posicionar el direccionamiento de la SRAM en punto inicial del cuadrado que marca las variables de entrada de la función.
   Dado que el direccionamiento esta codificado en versión X e Y, para una fila de 400 pixeles se necesitan 9 bits, por lo que al desplazar ese número de bits estamos posicionando la dirección en la fila deseada.

Si se modifica el valor, las filas no se cambiarán de forma adecuada y es posible que, según el valor, aparezcan trozos que deberían ir en otra fila en la misma o en filas posteriores.

Ejercicio 1: Control de Periféricos

4. ¿Qué pasa si queremos cambiar la configuración del códec?
   Si revisamos la documentación aportada para la licencia universitaria, observamos que es posible cambiar la configuración de dos formas:
   1. Cambiando la configuración del platform designer y volver a realizar la síntesis del sistema.
   2. Cambiar la configuración mediante software introduciendo el IP "Audio and Video Config".
5. ¿Hay que modificar los parámetros del componente en la Platform Designer y volver a compilar?
   No necesariamente. Aunque es posible.
6. ¿Se puede cambiar la configuración inicial por software?
   Es posible si se añade el "Audio and Video Config" al sistema mediante *platform designer.*
7. ¿Explica el procedimiento para modificar la configuración del códec por software?
   En la documentación nos indican que el procedimiento a seguir es:
   1. Poner en el registro de control el dispositivo apropiado en caso de que haya varios dispositivos.
   2. Escribir la dirección del control del periférico en el registro de direcciones.
   3. Escribir los datos que configuran el registro de control del periférico.
   4. La transferencia esta completa cuando el bit RDY en el registro de estado está a 1.
   5. El bit ACK será cero si no ha ocurrido ningún problema en la transferencia.
8. ¿Cómo se puede modificar el tiempo de grabación? ¿Cómo podemos grabar un mensaje de 15 segundos?
   La variable definida con "#define BUF_SIZE 500000" establece la longitud del buffer de grabación por lo que cambiando su valor podremos modificar la duración de la grabación.
   Si el ADC adquiere 48K muestras/seg se necesitaran alrededor 720K para obtener 15 segundos de grabación.

Ejercicio 2: El uso del HAL

9. ¿En que parte del código introduciríamos una instrucción que dibujase una línea horizontal en la pantalla MTL?

   Para dibujarla, se debería de escribir en el momento en el que se vuelve a dibujar el texto y el cuadrado (una vez se ha cambiado su dirección). De esta forma, la línea no se borrará cuando el cuadrado pase por encima.

10. ¿Cuál seria la instrucción a colocar?

    Mediante la instrucción HAL alt_up_pixel_buffer_dma_draw_hline () podemos dibujar la línea horizontal en la pantalla.

11. ¿Cómo se podría hacer para que la línea horizontal se desplazase a la misma velocidad que el cubo naranja recorriendo la pantalla de arriba abajo?

    Haciendo que la variable Y de la función cambie dentro de la instrucción "if (delay_MTL == 0)" tal y como se muestra en la siguiente figura.

```
y1++;
if (y1 >= alt_up_pixel_buffer_dma_y_res(pixel_buffer_dev_MTL) - 1)
{
    y1 = 0;
}

/* now draw a horizontal line for the example exercise completion */
alt_up_pixel_buffer_dma_draw_hline(pixel_buffer_dev_MTL, x1, x2, y1, 0x2345, 0);
```

Fig. 1. Snippet de código que implementa el desplazamiento de la línea horiontal a la misma velocidad que el cubo naranja de arriba abajo.

Cabe recordar, que es necesario borrar la línea (o dibujarla, pero del color del fondo) para no llenar la pantalla de líneas.

Ejercicio 3: El manejo de las Interrupciones

12. ¿Cómo se puede modificar la velocidad del patrón que se visualiza en los displays 7-segmentos?

    Cambiando la variable *"count"* para que el módulo del contador de 32 bits que incorpora nuestro sistema NIOS II sea diferente.

13. ¿Qué tenemos que cambiar para que la velocidad se ajuste a 1 segundo? ¿Es exacta la velocidad o aproximada?

    Para obtener 1 segundo necesitaríamos tener un valor de 50000000 (0x 2FAF080) puesto que la frecuencia del reloj del sistema es de 50MHz.

14. ¿Se podría realizar un gusanillo con velocidad ajustable por los pulsadores? ¿Cómo?

Un ejemplo, seria utilizar una variable global que actúa como *flag* que se activará cuando se ejecute la subrutina de la interrupción y cambiará el módulo del contador por el deseado (En nuestro caso un *ring* de 3 velocidades). En mi caso, el *main* sería algo parecido a lo de la siguiente figura (con los ajustes

```
//int counter = 0x2625a0;               // 1/(50 MHz) x (0x2625a0) = 50 msec
int speed[3] = {0xBEBC20, 0x2625a0, 0x2FAF080};             // 1/(50 MHz) x (0x2FAF080) = 1 sec
int index = 0;
*(interval_timer_ptr + 0x2) = (speed[1] & 0xFFFF);
*(interval_timer_ptr + 0x3) = (speed[1] >> 16) & 0xFFFF;

/* comienza el timer y habilita las interrupciones */
*(interval_timer_ptr + 1) = 0x7;    // STOP = 0, START = 1, CONT = 1, ITO = 1
alt_irq_register(TIMER_IRQ, NULL, interval_timer_isr);

//*(KEY_ptr + 2) = 0xE;         /* Mascara de los pulsadores (bit 0 es reset) */
*(KEY_ptr + 2) = 0xF;       /* Mascara de los pulsadores (bit 0 es cambio de velocidad) */
*(KEY_ptr + 3) = 0;
alt_irq_register(PUSHBUTTONS_IRQ, NULL, pushbutton_ISR);

while(1){
    *(HEX3_HEX0_ptr) = pattern;             // Visualiza el patr◆n en HEX3 ... HEX0
    *(HEX7_HEX4_ptr) = pattern;             // Visualiza el patr◆n en HEX7 ... HEX4
    if(Change_count_flag == 1)
    {
        Change_count_flag = 0;
        if (index < 2)
        {
            index = index + 1;
        }
        else
        {
            index = 0;
        }
        *(interval_timer_ptr + 0x2) = (speed[index] & 0xFFFF);
        *(interval_timer_ptr + 0x3) = (speed[index] >> 16) & 0xFFFF;
        *(interval_timer_ptr + 1) = 0x7;    // STOP = 0, START = 1, CONT = 1, ITO = 1
        printf("%d, %d \n", speed[index], index);
    }
}
```

Fig. 2. Snippet de código para implementar el cambio de velocidad del patrón de los 7-segmentos. Se tienen tres velocidades (2ms, 50ms, 1s) que se irán seleccionando en forma de loop a medida que se dispare la interrupción.

correspondientes en la subrutina de la interrupción).

15. ¿Qué pasa si todos los interruptores están abajo (a cero) y pulsamos el KEY3?

Pues que dado que el patrón de los 7-segmentos depende de los valores de los *switches*, no habrá ningún patrón que los 7-segmentos puedan mostrar por lo que estarán apagados.