Reloj Digital Con Alarma implementado en una FPGA

David Ricardo Martínez Hernández Código: 261931
Juan Pablo Rodríguez Rojas Código: 261744
Camilo Andrés Garzón Código:

***Resumen*— El reloj con alarma que implementamos es un reloj que funciona con hora definida de 0 minutos con 0 horas hasta 23 horas y 59 minutos, es decir se denomina una hora militar. Al inicializar el reloj automáticamente se muestra la hora del reloj que por defecto será las 0 horas con 0 minutos, mientras no se modifique la hora esa será la hora a la que el reloj se encuentre programado, para modificar esta hora podemos observar la Fig.1 que corresponde a los controladores de la FPGA en la cual se esta presentando la aplicación. Para activar la modificación de la hora se activa el primer switch de derecha a izquierda, aquel que se denomina $SW0$, luego se podrá aumentar los minutos con el primer pulsador de izquierda a derecha ($BTN0$), y para las horas se hace con el pulsador que se encuentra enseguida ($BTN1$), así se realiza la modificación de la hora del reloj. Para modificar la alarma se debe desactivar el switch $SW0$, y se procede a activar el switch que se encuentra en la primera posición de izquierda a derecha es decir $SW7$, se puede observar cambia la visualización del los displays $7-segmentos$, observando la hora en la que se encuentra programada la alarma (por defecto se encuentra programada a las 0 horas con 0 minutos). Luego después de activar $SW7$, se puede modificar la alarma utilizando los mismos pulsadores que usaron para modificar la hora del reloj. Para terminar se desactiva $SW7$ y en el momento en que coinciden la hora de la alarma con la hora del reloj se puede observar como se prende el primer Led de izquierda a derecha ($LD7$)) (Fig. 1), lo que indica que se activo la señal de alarma esta podrá ser desactivada si se presiona el pulsador que esta en la primera posición de derecha a izquierda ($BTN3$).**

***Palabras clave*— Flanco de bajada, Flanco de subida, Flip-Flops, Frecuencia, Horas, Minutos, Registros, Segundos, Señal de reloj.**

I. Objetivos

- Utilizar las estrategias de comprensión de análisis de sistemas digitales aprendidas en la clase para solucionar un problema cotidiano.
- Aprender la correcta utilización de los lenguajes HDL para la implementación de los sistemas digitales.
- Manejar los dispositivos programables FPGA para poder implementar más recursivamente, y por supuesto con el fin de facilitar la implementación del proyecto.
- Analizar como los diferentes tipos de descripción secuencial o combinacional, pueden ser utilizados con el fin de describir distintas tareas.

II. Introducción

Por un tiempo prolongado se programo en lenguajes como **FORTRAN**, **Pascal** y **C** en sus inicios se usaron para la descripción de programas de computadoras haciendo de estos secuenciales por naturaleza.
VHDL viene de **VHSIC** (*Very Speed Integrated Circuit*), VHDL es un lenguaje de descripción y modelado, diseñado para describir la funcionalidad y la organización de sistemas hardware digitales, placas de circuitos y componentes.
VHDL fue diseñado como en lenguaje para el modelado y simulación lógica de los sistemas digitales, además es un lenguaje con una sintaxis amplia y flexible permitiendo así el modelado estructural, teniendo como objetivo el desarrollo de un modelo para la **simulación** de un circuito. También permite el diseño Top-Down, permitiendo describir el comportamiento de los bloques de alto nivel, analizándolos y refinando la funcionalidad de alto nivel requerida antes de llegar a niveles más bajos de abstracción de la implementación del diseño[1].

Las **FPGA** (*Field Programmable Gate Array*) (Fig.1), introducidas por **Xilinx** en 1985, son el dispositivo programable por el usuario de más general espectro. También se denominan **LCA** (*Logic Cell Array*). Consisten en una matriz bidimensional de bloques configurables que se pueden conectar mediante recursos generales de interconexión. Estos recursos incluyen segmentos de pista de diferentes longitudes, más unos conmutadores programables para enlazar bloques a pistas o pistas entre sí. En realidad, lo que se programa en una **FPGA** son los conmutadores que sirven para realizar las conexiones entre los diferentes bloques, más la configuración de los bloques.

Fig. 1: Interruptores y Pulsadores FPGA Spartan3

Un ***decodificador*** es un circuito lógico con variables de varias entradas y salidas que convierte las entradas codificadas en salidas codificadas, donde los códigos de entrada son diferentes, en donde el código de entrada tiene generalmente menos bits que el de salida.[2]
Un ejemplo muy común de aplicación de los contadores son

[1] Texto tomada de [5]
[2] Definición tomada de [3], *Pág 351*

los sistemas de control de tiempo, entre los cuales se encuentra **El Reloj Digital**. El cual presenta la hora en los displays distribuidos de la siguiente manera (Fig. 2):

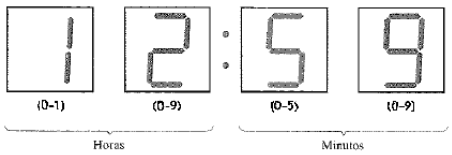

Fig. 2: Formato Hora [4], *Pág 594*

Se genera un tren de impulsos a 1 Hz para tener el control de los segundos, esta señal de reloj actualizara los estados de los contadores. Este circuito se puede implementar por medio de Flip-Flops conectados sincrónicamente.
Los contadores cuentan desde 0 hasta 59 y lugo vuelve a 0 para la etapa de *minutos* y *segundos*, para la etapa de *horas* se implementa por medio del **Contador de Décadas** contando desde 1 hasta 12, o de 0 hasta 23 de acuerdo a la configuración del fabricante.

III. Materiales y Métodos

Para realizar esta práctica se necesito:

- Computador.
- FPGA.
- Software Xilinx, Inc.

IV. Análisis y Resultados

Debido a que se implemento el sistema utilizando una descripción de modularidad, es necesario iniciar con el modulo central del programa que será el encargado de unir los demás submodulos que se crearon, este tipo de organización de los módulos en principales o top, y secundarios o Down, se denomina un diseño top Down, el cual es el nombre técnico que se le asigna a este tipo de descripción de tareas.
Como lo mencionamos utilizamos un modulo central que es el siguiente (Fig. 3):

Como se puede apreciar en el anterior código se hace llamado a los demás módulos, además se observa que existen muchas variables de tipo wire, estas variables funcionan de manera que sirven para conectar los diferentes módulos. Es así que iniciaremos explicando cada uno de los submodulos y paralelamente explicaremos como utilizamos las salidas de estos submodulos en la entra de los demás submodulos y así podremos explicar el funcionamiento del proyecto realizado.
Como vemos en el modulo central se enuncian las variables utilizadas, primero las entradas, que en este caso son los comandos que se encuentran entre las líneas 28 a 33, y así se encuentran referenciadas todas las variables que se utilizaron para la implementación del presente proyecto.
Como vemos los dos primeros módulos que utilizamos se denominan *debouncer* (nombre en ingles para anti-rebote), debido a que para modificar los minutos y las horas utilizamos pulsadores necesitamos crear un modulo de anti rebote para evitar este problema de los pulsadores, así que las salidas de

```
21  module RelojCentral(seg,vis,alarma,sonido,breloj,hora,minutos,off,clk
22      );
23
24      //////////////////////////////////////////////////
25      //Declaracion de las entradas
26      //////////////////////////////////////////////////
27
28      input clk;//Reloj de a FPGA a 50Mhz
29      input alarma;//Switch para modificar la alarma
30      input breloj;//Switch para modificar el reloj
31      input hora;//Boton para modificar las horas
32      input minutos;//Boton para modificar los minutos
33      input off;//boton para apagar la alarma
34
35      //////////////////////////////////////////////////
36      //Declaracion de las Salidas
37      //////////////////////////////////////////////////
38
39      output sonido;//Sonido de la alarma
40      output [3:0]seg;//4-7 Segmentos
41      output [6:0]vis;//Visualizacion del 7 segmentos
42
43      //////////////////////////////////////////////////
44      //Variable de interconexion entre modulos
45      //////////////////////////////////////////////////
46
47      wire dh;//Boton de horas sin rebote
48      wire dm;//Boton de minutos sin rebote
49      wire fmin;//frecuencia de un minuto
50      wire fmux;//frecuencia del mux
51
52      wire [3:0] h0;//unidades de minutos alarma
53      wire [3:0] h1;//decenas de minutos alarma
54      wire [3:0] h2;//unidades de hora alarma
55      wire [3:0] h3;//decenas de hora alarma
56
57      wire [3:0] i0;//unidades de minutos reloj
58      wire [3:0] i1;//decenas de minutos reloj
59      wire [3:0] i2;//unidades de hora reloj
60      wire [3:0] i3;//decenas de hora reloj
61
62      wire [3:0] j0;//unidades de minutos
63      wire [3:0] j1;//decenas de minutos
64      wire [3:0] j2;//unidades de hora
65      wire [3:0] j3;//decenas de hora
66
67      wire mala;//estimulo unidades de minutos de alarma
68      wire hala;//estimulo unidades de horas de alarma
69      wire mrel;//estimulo unidades de minutos de reloj
70      wire hrel;//estimulo unidades de horas de reloj
71
72      wire ifreqmin;//frecuencia de 1 minuto de salida del modulo reloj
73
74      wire [3:0] numerofinal;//numero para la multiplexacion de los 7-seg
75
76      //////////////////////////////////////////////////
77      //Proceso
78      //////////////////////////////////////////////////
79
80      debouncer modulo1(dm,minutos,clk);
81      debouncer modulo2(dh,hora,clk);
82
83      freq1 modulo3(fmin,clk);
84      freq2 modulo4(fmux,clk);
85
86      muxbutones modulo5(mala,hala,mrel,hrel,dm,dh,breloj,alarma);
87
88      fsmalarm modulo6(h0,h1,h2,h3,mala,hala);
89
90      fsmreloj modulo7(i0,i1,i2,i3,ifreqmin,breloj,hrel,mrel,fmin);
91
92      muxnums modulo8(j0,j1,j2,j3,i0,i1,i2,i3,h0,h1,h2,h3,alarma);
93
94      sound modulo9(sonido,off,ifreqmin,h0,h1,h2,h3,i0,i1,i2,i3);
95
96      muxvisual modulo10(numerofinal,seg,j0,j1,j2,j3,fmux);
97
98      visualizacion modulo11(vis,numerofinal);
99
100
101
102 endmodule
```

Fig. 3: Código utilizado

este modulo son dm y dh, que serán los nuevos impulsos pero que tendrán evitado dicho inconveniente.
Los siguientes módulos son freq1 y freq2, ambos son dos módulos que dividen la frecuencia del reloj interno de la FPGA que estamos utilizando que tiene una frecuencia de $50\ MHz$, el primer modulo crea una señal de reloj que tiene un periodo de 1 minuto, $fmin$, el cual ha de ser utilizado para modificar automáticamente la hora del reloj. Mientras que el segundo entrega una señal de reloj que tiene una frecuencia de $200\ Hz$, $fmux$, que hemos de utilizarla para la visualización de la hora en los cuatro displays $7 - segmentos$, proceso que explicaremos posteriormente.
El siguiente modulo que utilizamos se denomina $muxbotones$, que como su nombre insinúa se trata de un multiplexor, que debido a que los botones que modifican manualmente la hora y los minutos del reloj y de la alarma son los mismos, se necesitaba un sistema que decidiera en el momento en que se pulsan dichos botones que le asignara el cambio a lo que se quiere modificar ya sea el reloj o la alarma, es así que por ende de este modulo existen cuatro salidas, el estimulo de las horas del reloj, $hrel$, el de minutos del reloj, $mrel$, el impulso que genera las modificaciones de la alarma en cuanto a minutos, mala, y el que genera la modificación de las horas de la alarma, hala.
El siguiente modulo que utilizamos es el modulo $fsmalarm$, es el modulo que genera la modificación de la alarma, y como vemos genera cuatro salidas $h0$, $h1$, $h2$, $h3$, que son números de $4 - bits$, deben de ser de ese tamaño puesto que para la parte de decodificación para generar el numero mas alto que tenemos que en este caso es 9, se necesitan $4 - bits$, donde $h0$, significa las unidades de minutos, como podemos apreciar en la declaración de las variables para este modulo líneas de 52 a 55.
El siguiente modulo que utilizamos es $fsmreloj$, el cual como en el caso anterior se encarga igualmente de modificar el reloj de la manera que describimos en el proceso de funcionamiento, igualmente como en el modulo anterior genera cuatro números de $4 - bits$, que son los que se muestran en las líneas 57 a 60, pero este modulo además genera una señal cuadrada de un minuto llamada $ifreqmin$, la cual se explicara su funcionamiento en un modulo posterior.
El siguiente modulo es $muxnums$, como su nombre revela se trata de nuevo de un multiplexor, el cual cumple la función de escoger entre los números del reloj o los de la alarma, ya que en cuanto se modifica el switch que activa la opción de cambiar la hora del reloj, inmediatamente en la visualización aparece la hora de la alarma, por eso necesitamos escoger entre los números que se visualizaran, y por ende las salidas de este modulo son $j0$, $j1$, $j2$, $j3$, que están definidas en las líneas 62 a 65.
Luego sigue el modulo sound, que es el encargado de comparar el valor de la hora del reloj con la de la alarma y generar un estimulo que será el Led en caso de que coincidan ambas horas, así como también esta encargado de controlar que cuando se desee apagar la alarma se pueda hacer, como vemos este modulo tiene como entradas los numero de la alarma, así como los números dl reloj, el botón off que funciona para desactivar la alarma , y la frecuencia de $ifreqmin$, que es una señal de 1 minuto de periodo que salía del modulo que modifica la hora del reloj.
La frecuencia de este modulo aunque es la misma que la señal de un minuto de periodo tiene cierto retraso ya que se activa apartir de la señal de un minuto de periodo inicial, y por ende permite que se comparen los números en este modulo inmediatamente apenas cambia, ya que de no ser así se activaría la alarma siempre un minuto después del momento en el cual debería activarse, esto funciona debido a que como los minutos cambian en el flanco de subida de la señal original de 1 minuto, entonces aunque sean iguales no seria si no hasta el siguiente flanco que el modulo anterior compararía y generaría la señal de alarma, por ende al hacer este cambio aunque seguimos trabajando con una señal de 1 minuto de periodo esta ocurre después de la original, por ende el flanco de subida será un tiempo posterior, pudiendo entonces comparar los números en el mismo instante en que cambian y así generar la señal de alarma en el momento adecuado.
El siguiente modulo que necesitamos es el modulo $muxvisual$, lo que genera este modulo es que debido a que tenemos que cada 7 segmentos es un ánodo diferente, en el caso de que no se asigne un ánodo especifico, todos los $4 - 7\ segmentos$ mostraran el mismo numero, por ende necesitamos que cada uno muestre un numero diferente la mayoría del tiempo, por ende realizamos la función de que fuera apagando 3 ánodos cada vez y mostrando en el ánodo que queda encendido el numero deseado.
Por eso es que para que el ojo humano no detecte que se esta pagando y prendiendo ánodos, es que se hace esto a una frecuencia de $200\ Hz$, la cual salía del modulo $freq2$, así es que realizamos el proceso que generara una visualización que el ojo detectara como si fueran todos ánodos diferentes, y por ende es que las entradas de este modulo son las que salían del modulo $muxnums$, descrito anteriormente, y como debemos ver un numero en cada 7 segmentos por eso es que la salida de este modulo, aparte de los $4 - 7$ segmentos, es un numero denominado $numerofinal$, el cual es utilizado en la etapa del siguiente modulo.
Y para finalizar la descripción de los módulos y del funcionamiento del reloj, tenemos el modulo de visualización que consiste en un codificador binario BCD, que se utiliza para el 7 segmentos de ahí que la salida sea un numero de 7-bits que es el necesario para hacer la decodificación, y así es como podemos ver un numero BCD en los segmentos.

V. Conclusiones

- Se comprendió que la utilización de los lenguajes para describir hardware funcionan de manera paralela y no secuencia aunque el lenguaje es muy parecido a $C++$ el cual funciona de manera secuencial.
- El lenguaje HDL o $VHDL$ es una herramienta muy útil para diseñar y modelar sistemas de hardware, placas de circuitos y componentes, aunque su principal función es realizar simulaciones muy precisas , también tiene la ventaja que se puede escribir muy similar a C, es decir posee un lenguaje de programación muy conocido y básico.

- El principal problema al realizar esta práctica fue pasar el $carri$ de salida, es decir cuando el reloj llegara a 59 minutos tenia que hacer el cambio de hora (valor de la hora mas 1) pero no lo realizaba, debido a que se encontraba una posición que lo llevaba a 00 y no contaba ese 1.
- Se realizaron muchas pruebas, lo primera que se realizo fue comprobando que estaba contando pero contaba de 0 a F, luego se modifico el código (se encontraba como una maquina de estados) a un diseño muy parecido a C. Finalmente se realizaron las pruebas correspondientes pero no funcionaba el $carri$ de salida, al solucionar este problema se obtuvo lo que se esperaba.

REFERENCIAS

[1] Dorf Svoboda. *"'Circuitos Eléctricos"'*. Alfaomega, 2006.
[2] C. J. Savant. *"'Diseños Electrónicos: Circuitos de Sistema"'*. Prentice-Hall, 2006.
[3] John F. Wakerly. *"'Diseño Digital: Principios y Prácticas"'*. Prentice-Hall, 2001.
[4] Thomas L. Floyd. *"'Fundamentos de Sistemas Digitales"'*. Prentice-Hall, 2000.
[5] Sito Web: http://www.jeuazarru.com/docs/VHDL.pdf