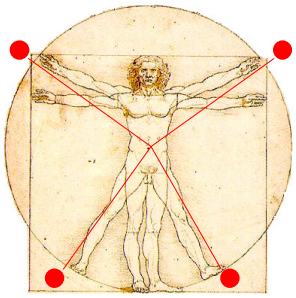
TECNOLOGÍ@ y *DESARROLLO*

Revista de Ciencia, Tecnología y Medio Ambiente

VOLUMEN III. AÑO 2005

SEPARATA



INTERFAZ VISUAL PARA LA SIMULACIÓN DE UN CONTROLADOR HARDWARE DE NIVEL DE LÍQUIDO EN UN DEPÓSITO MEDIANTE LA HERRAMIENTA TCL/TK

Juan Suardíaz Muro, Basil M. Al-Hadithi

UNIVERSIDAD ALFONSO X EL SABIO
Escuela Politécnica Superior

Villanueva de la Cañada (Madrid)

ISSN: 1696-8085
Editor: Julio Merino García tecnologia@uax.es

INTERFAZ VISUAL PARA LA SIMULACIÓN DE UN CONTROLADOR HARDWARE DE NIVEL DE LÍQUIDO EN UN DEPÓSITO MEDIANTE LA HERRAMIENTA TCL/TK

Juan Suardíaz Muro[1], Basil M. Al-Hadithi[2]

[1)]Dr Ing. Industrial,
Departamento de Tecnología Electrónica, Escuela Técnica Superior de Ingenieros Industriales,
Universidad Politécnica de Cartagena (Murcia).
Campus Moralla del Mar, 30202, Cartagena. España. Tlf.:968325380, email: Juan.Suardiaz@upct.es

[2)]Dr Ing. Industrial,
Departamento de Electrónica y Sistemas, Escuela Politécnica Superior, Universidad Alfonso X el Sabio.
Avda. De la Universidad nº1, Villanueva de la Cañada, 28691 Madrid. España. Tlf.:918105035, email:
bmal@uax.es

RESUMEN: Este artículo se presenta el desarrollo de una interfaz visual, basada en el lenguaje Tcl/Tk, la cual permita una fácil y sencilla verificación de la funcionalidad de núcleos hardware implementados con el lenguaje de descripción hardware VHDL, evitando así la tediosa forma de validación asociada al análisis de cronogramas en entornos clásicos de simulación

PALABRAS CLAVE: Lenguajes de programación Tcl/Tk, Controlador de Nivel de Líquido, FPGAs, Arquitecturas Reconfigurables

ABSTRACT: This paper deals with a novel approach of visual validation of hardware designs, based on the Tcl/Tk programming language. It will be shown with an illustrative example of the design of a liquid level controller how this language becomes a powerful tool for checking the design's functionality. Thanks to the possibility of a low level communication between the Tcl/Tk visual interface and the classical ModelSim VHDL simulatio, the usually and normally tedious validation method of analyzing timing diagrams is avoided.

KEY-WORDS: Tcl/Tk Programming Language, Liquid Level controller, FPGAs, Reconfigurable Architectures

1. Introducción

Tcl (*Tool Command Language*) fue creado, en un principio, como un instrumento para unir varios modulos de software en una aplicación por John K. Ousterhout y su equipo de la Universidad de California. Actualmente es desarrollado por Sun Microsystems Laboratories y distribuido de forma gratuita (Venkat, 1999). Tcl puede funcionar en varias plataformas como Windows9x, Windows NT, WindowsXP, Macintosh y Unix funcionando con el sistema X Window.

Existen varios programas no compilados como son Perl y Python, aunque Tcl ha resultado más popular y puede encontrarse en la web numerosas páginas con aplicaciones; lo que ha hecho que Tcl haya adquirido una fuerte comunidad de usuarios y una mayor madurez. Además de esto, Tcl resulta el método más eficaz de integrar varias aplicaciones en un solo programa.

Tk (*Tool Kit*) es una extensión del lenguaje de programación Tcl dedicada a que el programador pueda ser capaz de crear una interfaz gráfica de usuario (GUI) de forma sencilla. Tk contiene comandos que posibilitan la creación de botones, menus, ventanas, etc. En Tk se denomina a estos elementos que permiten al usuario visualizar y modificar el estado del programa '*widgets*'.

o *Tcl/Tk y ModelSim SE 5.5f.*

Si bien la aplicación ModelSim se trata de un entorno de desarrollo (IDE, *Integrated Development Environment*) para los lenguajes de descripción hardware Verilog(ModelTech, 2005) y VHDL(IEEE, 1994), presenta embebido en su interior un núcleo Tcl/Tk que posibilita la creación de aplicaciones en las que la simulación hardware puede comunicarse con una interfaz gráfica basada en Tcl/Tk.

En tiempo de simulación hardware, es posible usar comandos de ModelSim, primero para comenzar la simulación y después para cambiar los valores de las entradas y para observar los valores de las salidas, los cuales también se mostrarán de forma gráfica gracias a la comunicación con la interfaz Tcl/Tk.

Así pues, primeramente se utilizará el comando '*vlib*' en la consola de ModelSim para la creación de una librería, necesaria para la posterior compilación del programa en VHDL

(ModelTech, 2005b). Será utilizado para crear la librería Work del siguiente modo: '*vlib work*'. Esta librería debe ser creada en el mismo directorio en el que se encuentra el programa en VHDL. Una vez creada la librería se pasa a compilar el programa en VHDL, tras lo cual es posible simularlo con el comando '*vsim*', escribiendo simplemente en la consola de ModelSim la palabra '*vsim*' y dentro del menú que aparece seleccionando el archivo que se desea simular.

Tras iniciar la simulación, se utilizarán los comandos que permiten variar los valores de las señales de entrada y acceder a los valores que van tomando las señales de salida. Estos comandos serán los incluidos en el programa en Tcl/Tk para controlar la simulación, cambiando los valores de las señales de entrada y visualizando los valores de las señales de salida. La instrucción '*force*' también permite al usuario variar el estado de una señal en ModelSim a continuación. De esta forma, el comando '*force nombre_señal 1*' consigue establecer el valor de la señal nombre_señal a 1.

En el ejemplo de la figura 1.1 aparece la consola de Modelsim y una ventana en la que es posible encontrar la señal de reloj (clk) y la señal f1 asociadas a un módulo controlador de teclado, desarrollado en VHDL y denominado '*teclado*'. Primero mediante la ejecución de la instrucción '*force f1 1*' en la consola de ModelSim se pone a 1 la señal f1. Para comprobar el cambio en la señal se utiliza el comando '*examine –value f1*' y se comprueba que el cambio de estado se produce después de que se realice una continuación en la simulación con la instrucción '*run*'.

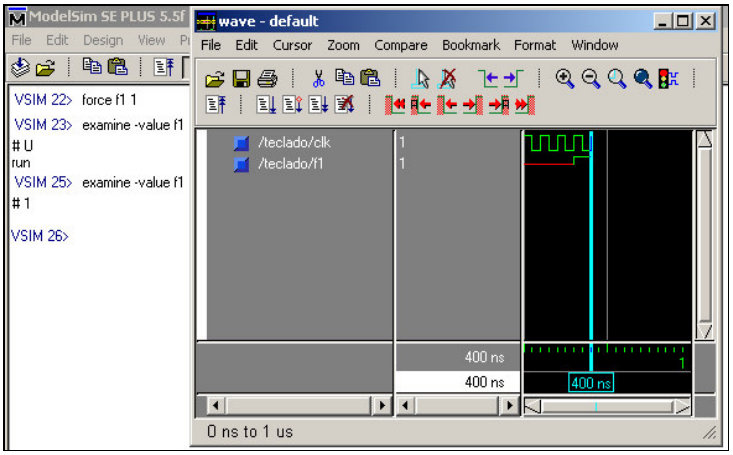


Fig. 1.1 Consola de ModelSim con visualizador de formas de onda.

2. Componentes del sistema de control de nivel de un líquido

En este artículo se van a tratar las características del control de nivel de líquido de un depósito. En la figura 2.1 se observa el esquema del depósito con dos válvulas una para llenado (de control) y otra para vaciado (de carga). Además de estos elementos, también dispone de tres sensores de nivel para controlar el nivel de llenado del depósito. Cabe destacar que las dos válvulas disponen de un selector de nivel de apertura que regula la velocidad de llenado y vaciado de forma independiente. Este nivel de apertura de las válvulas es seleccionado mediante dos barras que incluye el programa. El funcionamiento básico del programa en VHDL es el de mantener el nivel de liquido del depósito entre los niveles adecuados valiéndose de la información de los sensores y activando y desactivando tanto la válvula de control como la válvula de carga.

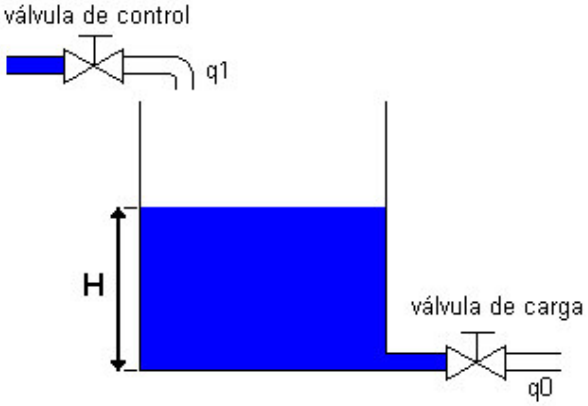


Fig. 2.1 Esquemático del depósito a controlar.

El programa en TCL/TK tiene la función de representar gráficamente el depósito, el nivel de llenado, actuar sobre el nivel de apertura de las válvulas y enviar la información de los sensores a la simulación en Modelsim. Más adelante se tratarán en detalle las características de ambos programas.

Como ya se ha comentado, los elementos que forman el proceso que se desea controlar son el deposito de líquidos, tres sensores de nivel de líquidos, dos electro válvulas y la FPGA. La FPGA se utilizará para recibir como entradas las señales procedentes de los tres sensores de nivel y para controlar la apertura y cierre de las dos válvulas.

El algoritmo de control implementado sobre la FPGA se puede resumir de la forma siguiente:

- En condiciones de llenado normal, las válvulas E y S se encuentran abiertas.
- Si el líquido llega al nivel de vacío, se cierra la válvula de salida y se mantiene abierta la de entrada.
- Si el líquido llega al nivel de lleno, se cierra la válvula de entrada y se mantiene abierta la de salida.
- Si por cualquier circunstancia, por ejemplo lluvia, se llegara al nivel de alarma, se deberá cerrar la válvula de entrada y abrir la de salida. Esta situación se mantendrá hasta que el tanque llegue al estado de vacío.

3. Implementación del algoritmo de control sobre la FPGA

Es posible representar dicho comportamiento mediante la máquina de estados de la figura 3.1.

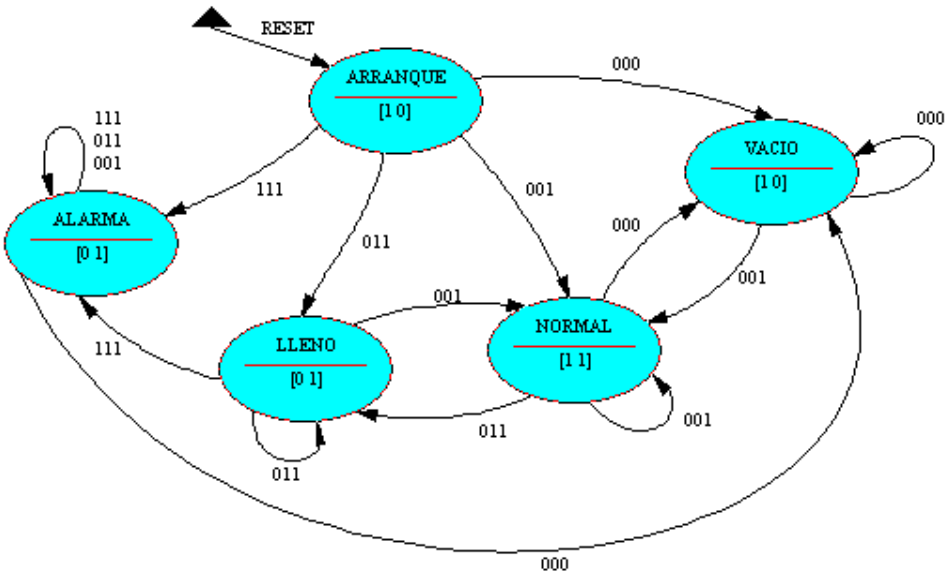


Fig. 3.1 Máquina de estados asociada al algoritmo de control.

Como es posible deducir del diagrama de estados anterior, se ha optado por una versión basada en una máquina de Moore, la cual presenta la ventaja de minimizar la lógica de

salida del circuito y en este caso conduce a un conjunto de estados que de forma intuitiva los alumnos pueden asociar a los diferentes niveles del depósito.

Una vez encontrado el diagrama de estados asociado al algoritmo de control, su traducción al lenguaje de descripción de hardware VHDL es inmediata, siendo su código asociado el listado a continuación:

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY MaquinaEstadosMoore IS
  PORT ( sA, sB, sC: IN std_logic;      --Señal de entrada.
         CLK: IN std_logic;             --Señal de reloj.
         sResetH: IN std_logic; --Señal de inicialización
         sE, sS: OUT std_logic);   --Salida
END MaquinaEstadosMoore;

ARCHITECTURE MaquinaEstadosMooreArch OF MaquinaEstadosMoore IS
  --Declaración del tipo asociado a los estados.
    TYPE TipoEstados IS (VACIO, NORMAL, LLENO, ALARMA);
  --Señales auxiliares para la codificación del estado actual y siguiente.
    SIGNAL tEstadoActual, tEstadoSiguiente: TipoEstados;
BEGIN
-- Proceso dedicado a la lógica de estado:
LOGICA_ESTADO: PROCESS(tEstadoActual, sEntrada)
BEGIN
CASE (tEstadoActual) IS
  WHEN VACIO =>
    IF (sA = '0' and sB='0' and sC = '0') THEN tEstadoSiguiente <= VACIO;
    ELSIF (sA = '0' and sB='0' and sC = '1') THEN tEstadoSiguiente <= NORMAL;
    ELSIF (sA = '0' and sB='1' and sC = '1') THEN tEstadoSiguiente <= LLENO;
    ELSIF (sA = '1' and sB='1' and sC = '1') THEN tEstadoSiguiente <= ALARMA;
    END IF;
  WHEN NORMAL =>
    IF (sA = '0' and sB='0' and sC = '0') THEN tEstadoSiguiente <= VACIO;
    ELSIF (sA = '0' and sB='0' and sC = '1') THEN tEstadoSiguiente <= NORMAL;
    ELSIF (sA = '0' and sB='1' and sC = '1') THEN tEstadoSiguiente <= LLENO;
    END IF;
  WHEN LLENO =>
    IF (sA = '0' and sB='0' and sC = '1') THEN tEstadoSiguiente <= NORMAL;
    ELSIF (sA = '0' and sB='1' and sC = '1') THEN tEstadoSiguiente <= LLENO;
    ELSIF (sA = '1' and sB='1' and sC = '1') THEN tEstadoSiguiente <= ALARMA;
    END IF;
  WHEN ALARMA =>
    IF (sA = '0' and sB='0' and sC = '0') THEN tEstadoSiguiente <= VACIO;
```

```
    ELSE tEstadoSiguiente <= ALARMA;
    END IF;
END CASE;
END PROCESS LOGICA_ESTADO;
-- Proceso dedicado a la Memoria de Estado
MEM_ESTADO: PROCESS(CLK, sResetH, tEstadoSiguiente)
BEGIN
--Inicialización con RESET_H
 IF (sResetH ='1') THEN tEstadoActual<=VACIO;
 ELSIF(rising_edge(CLK)) THEN tEstadoActual <= tEstadoSiguiente;
 END IF;
END PROCESS MEM_ESTADO;

--Zona concurrente dedicada a modelar la
--lógica de salida.
sS <= '0' WHEN (tEstadoActual = VACIO)
        ELSE '1';
sE <= '1' WHEN (tEstadoActual  = NORMAL or tEstadoActual  = VACIO)
        ELSE '0';

-- sSalida = f(Estado) => Máquina de MOORE.
```

END MaquinaEstadosMooreArch;

En la simulación, se parte del estado inicial con el depósito totalmente vacío (estado VACÍO), se habilita la válvula de entrada y deshabilita la de vaciado, de esta forma comienza a llenarse. Este proceso de llenado continúa hasta que se activa el primer sensor de nivel (sC), en este momento se pasa al estado NORMAL activándose entonces ambas válvulas. Si por alguna circunstancia externa (lluvia) subiera el nivel hasta alcanzar el sensor sB, se pasaría al estado de LLENO, y se cerraría la válvula de llenado manteniéndose la otra abierta. Si se activa el sensor sA el sistema pasará a el estado de ALARMA y las válvulas se comportarán de forma similar al estado LLENO.

Para controlar los cambios de estado se utilizan unas señales auxiliares tEstadoActual y tEstadoSiguiente. En el proceso LOGICA_ESTADO mediante un bucle 'case' se identifica el valor de la señal tEstadoActual. Una vez identificado se comprueban con bucles 'if' anidados el estado de los sensores (mediante las señales sA, sB y sC) y se da a la señal tEstadoSiguiente el valor correspondiente.

Para controlar el paso del estado siguiente a estado actual se realizará el proceso MEM_ESTADO así que mediante la utilización de un bucle 'if' si la señal sResetH ='1' se da a la señal tEstadoActual el valor del estado inicial (VACIO). Cuando se produce el

flanco de subida de la señal de reloj, 'rising_edge(CLK)', entonces tEstadoActual se actualiza con el valor de tEstadoSiguiente.

La última parte del programa es la encargada de establecer los valores de las salidas sS y sE que controlarán el estado de las válvulas de entrada y salida. Para conseguir esto establecemos que sS sólo se ponga a cero cuando tEstadoActual = VACIO y sE se ponga a 1 cuando tEstadoActual = NORMAL ó tEstadoActual = VACIO.

4. Diseño del modelo de simulación del depósito.

Con la herramienta TCL/Tk se ha desarrollado una interfaz visual que permite por una parte mostrar gráficamente el estado del llenado depósito y por otra interactuar con la herramienta de simulación de Modelsim, mandándole los datos de los sensores y recibiendo de esta el estado de apertura o cierre de las válvulas generado por el controlador implementado en VHDL.

Uno de los problemas fundamentales al afrontar esta tarea es el que surge a la hora de representar la variación en el nivel de agua dependiendo del estado de las válvulas. Para representar esta variación es necesario conocer un modelo matemático del depósito en el que introduciendo el estado de las válvulas y el nivel de agua se obtuviera como resultado el nivel actual.

En la figura 4.1 puede observarse el esquema del depósito, donde representamos las válvulas de entrada y salida, con sus caudales de entrada y salida asociados (qe y qs). Además de esto las válvulas tienen una restricción de salida que hay que tener en cuenta.

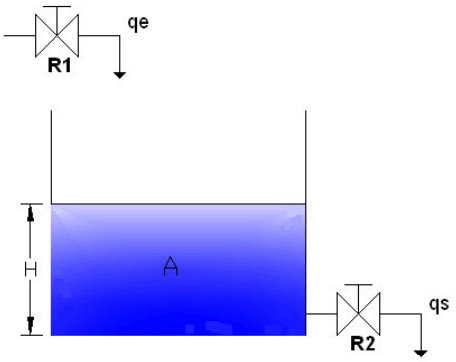


Fig. 4.1 Esquemático del depósito y variables de control.

Los otros dos parámetros a introducir en la ecuación son la capacidad hidráulica A (área de la base del depósito) y H la altura del nivel de líquido.

De esta forma en un intervalo de tiempo dt se puede considerar constantes qe y qs, por lo que el caudal resultante será (qe – qs) y el líquido acumulado en ese tiempo (qe-qs)*dt.

La acumulación de este líquido provocará un pequeño incremento en la altura (dh). El líquido acumulado se puede obtener también multiplicando el incremento de la altura por el área. Obteniéndose:

$$(qe - qs) \cdot dt = A \cdot dh$$

y dividiendo por dt:

$$(qe - qs) = A \cdot \frac{dh}{dt}$$

teniendo en cuenta que qs depende de la altura h que tenga el depósito en un momento dado y de la restricción R a la salida del depósito:

$$qs = \frac{h}{R}$$

Sustituyendo el valor de qs en la ecuación obtenemos como resultado:

$$qe - \frac{h}{R} = A \cdot \frac{dh}{dt}$$

$$dh = \left(\frac{qe}{A} - \frac{h}{R \cdot A} \right) \cdot dt$$

Teniendo en cuenta que el incremento de altura es igual a h – ho:

$$dh = h - ho$$

Y considerando el incremento de tiempo igual a uno, obtenemos la siguiente ecuación que se utilizará posteriormente en el programa:

$$h - ho = \left(\frac{qe}{A} - \frac{h}{R \cdot A} \right) \cdot dt$$

Despejando h:

$$h = ho + \left(\frac{qe}{A} - \frac{ho}{R \cdot A} \right) \cdot dt$$

El resultado es la interfaz gráfica representada en la figura 4.2, donde la altura del nivel del depósito viene regida por la ecuación anterior y es posible parametrizar en tiempo de simulación los valores asociados a las restricciones de entrada de las válvulas de entrada y salida.

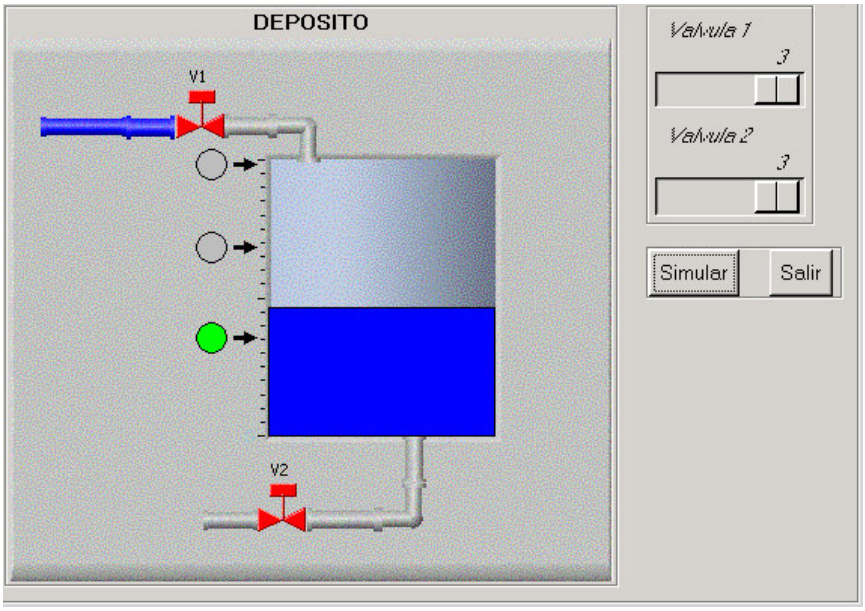


Fig. 4.2 Aspecto gráfico de la interfaz desarrollada con TCL/Tk.

Cuando se ejecuta la simulación, esta interfaz se comunica con el modelo VHDL del controlador a través de la herramienta ModelSim y se produce una actualización del nivel del depósito en cada instante, así como del estado de cada una de las válvulas, lo cual puede encontrarse reflejado en la figura 4.3.

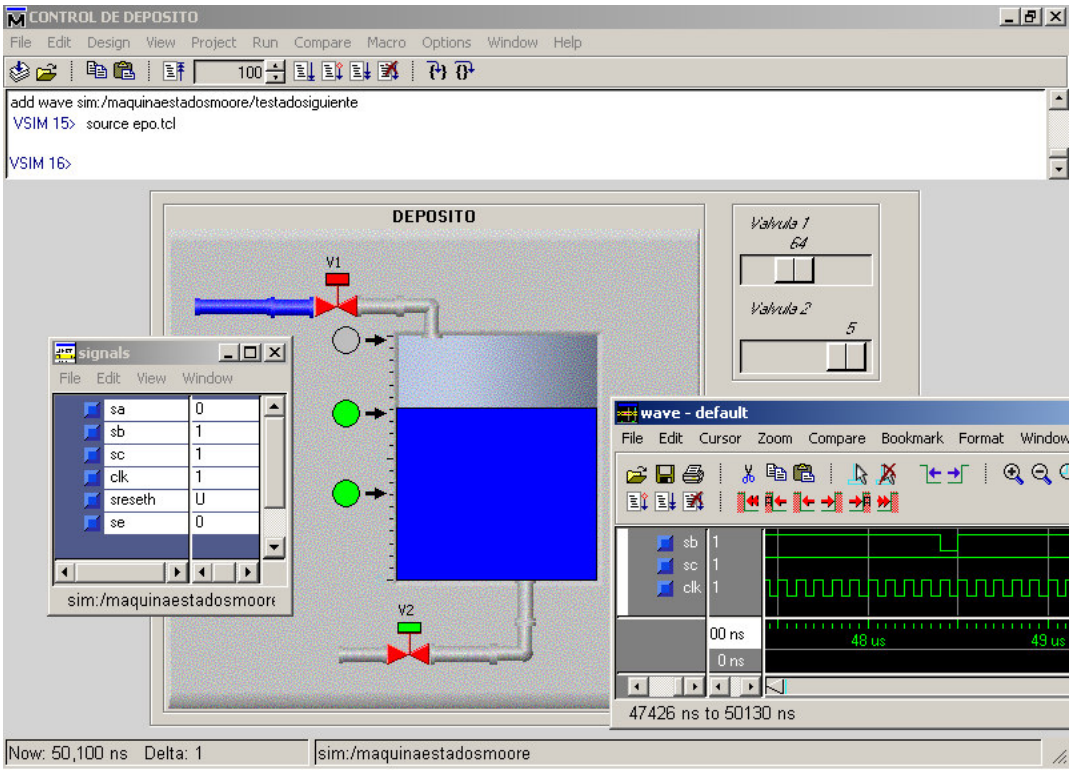


Fig. 4.3 Ejemplo de funcionamiento del entorno de simulación.

Como puede verse en la figura 4.3. los círculos representan los sensores, colocándose en color verde cuando el nivel de agua los supera y gris cuando no ha llegado. También puede verse el estado de apertura y cierre de las válvulas en el cuadrado del actuador siendo verde cuando esta activa la válvula y rojo cuando esta desactivada.

5. Conclusiones

La principal ventaja de la herramienta Tcl/Tk en conjunción con la herramienta de simulación ModelSim es que permite el desarrollo de una serie de prácticas en las que ya no es necesario que la comprobación del funcionamiento del núcleo hardware desarrollado se base en el análisis de las formas de onda proporcionadas por un entorno

más o menos amigable, sino que se ha conseguido que el interfaz de usuario sea similar a los componentes físicos que formarían realmente el sistema.

Se ha constatado que el desarrollo de controladores se ve facilitado por el empleo de esta herramienta visual, que de una forma amigable permite una rápida depuración en el desarrollo de los algoritmos de control.

Gracias a la posibilidad de cómputo analógica que posee el motor interno de Tcl/Tk es posible incrementar el número de sensores, lo que conduciría a una máquina de estados mayor, o incluso el sustituir los sensores discretos por unos que permitan una medición analógica del nivel del depósito en cada instante. Está previsto el desarrollo de un sistema basado en lecturas analógicas del nivel de líquido, en lugar de las provenientes de un número determinado de sensores de posición, lo que posibilitaría el desarrollo de un conjunto de controladores con una base matemática más compleja, como podrían ser controladores basados en estructura variable o incluso lógica fuzzy. El principal inconveniente de esto radica en el hecho de que para poder implementar estos algoritmos de control es necesario la utilización de operaciones en punto fijo, lo que complica el código a desarrollar o requiere la utilización de otras herramientas de diseño y simulación hardware, como pueda ser el entorno Matlab System Generador, creado gracias a la cooperación de las compañías Xilinx (Xilinx, 2005) y Mathworks (MathWorks, 2005).

Otra mejora que sí cae dentro del ámbito docente de la asignatura y en la que se está trabajando en la actualidad, es el desarrollo de controladores discretos utilizando el modelo en transformada z de versiones continuas, lo cual también es posible gracias a la enorme versatilidad que posee la herramienta.

Referencias

IEEE, (1994): *The IEEE Standard VHDL Language Reference Manual*, ANSI/IEEE-Std-1076-1993.

MATHWORKS (2005): The Matlab company http://www.mathworks.com

MODELTECH (2005): Start Here for ModelSim SE. Model Technology. Mentor Graphics Company.

MODELTECH (2005b): ModelSim SE Tutorial. Model Technology. Mentor Graphics Company.

TEXAS INSTRUMENT (1998): *SN7448 BCD to seven-segment decoder/driver* datasheet. Texas Instrument Inc.

VENKAT V.S.S. Sastry y Lakshmi Sastry (1999): *Tcl/Tk in 24 hours*. Sams Publishing.

XILINX (2005): The xilinx company http://www.xilinx.com