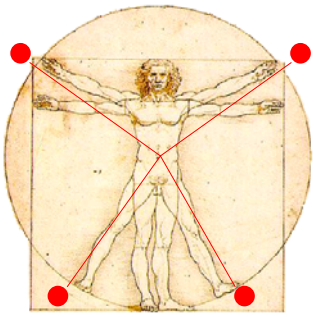
TECNOLOGÍ@ y *DESARROLLO*

Revista de Ciencia, Tecnología y Medio Ambiente

VOLUMEN IV. AÑO 2006

SEPARATA



DISEÑO DE UN SISTEMA DE BÚSQUEDA DE RUTA OPTIMA BASADA EN UNA INTERFAZ VISUAL 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)

DISEÑO DE UN SISTEMA DE BÚSQUEDA DE RUTA ÓPTIMA BASADA EN UNA INTERFAZ VISUAL MEDIANTE LA HERRAMIENTA TCL/TK

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

[1)]Dr. Ingeniero 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 Ingeniero 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: En este artículo se pretende diseñar un sistema de búsqueda de una ruta óptima que permite a los dueños de las empresas de ventas a domicilio la elección la ruta más corta para sus vendedores con el fin de reducir los gastos excesivos en concepto de vales de combustible y ticket de comidas.

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 VHDL, Lenguajes de programación Tcl/Tk, Controlador de ruta óptima, FPGAs, Arquitecturas Reconfigurables

ABSTRACT: In this paper a search system for an optimum path is presented. The optimum path allows sales establishments to find the shortest path for their sales to reach their customers. This will in turn reduces the costs of transportation. A novel way of visual validation of hardware designs, based on the Tcl/Tk program language is implemented.This language is proving to be 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 simulation the usually and normally tedious validation method of analyzing timing diagrams is avoided.

KEY-WORDS:VHDL Programming Language, Tcl/Tk Programming Language, optimum path system, FPGAs, Reconfigurable Architectures

1. Introducción

Tcl (*Tool Command Language*) es un lenguaje de programación interpretado y multiplataforma. Fue creado por John K. Ousterhout (Ousterhout, 2005) (Ousterhout, 1994) y su equipo de la Universidad de California, pero actualmente es desarrollado por Sun Microsystems Laboratories (en concreto por su grupo SunScript, que lidera el propio Ousterhout) y distribuido de forma totalmente gratuita, aunque su uso sea para aplicaciones comerciales, a través de Internet (SunLabs, 2005).

Tcl consiste fundamentalmente en un lenguaje de comandos, cuyo intérprete recibe el nombre de tclsh (tclsh80 para Tcl/Tk 8.0). Tiene como una de sus principales características la gran facilidad con la que se pueden implementar funciones en C/C++ que pasan a ser nuevas instrucciones del intérprete. Es decir, se pueden embeber aplicaciones en C/C++ dentro del propio intérprete de Tcl; de esta forma es posible obtener nuevas versiones de Tcl, denominadas extensiones, que no dejan de ser otra cosa que intérpretes que aúnan a los comandos originales de Tcl nuevos comandos escritos en C/C++.

Algunas de estas extensiones son BLT (que permite hacer representaciones gráficas en 2D), Itcl (Incremental Tcl, Tcl orientado a objetos), OraTcl (Tcl capaz de manejar bases de datos ORACLE), etc.

Pero sin duda, la extensión más conocida, y que es distribuida junto con el propio Tcl, es Tk (Tool Kit). Tk, creada por el propio John Ousterhout, proporciona un intérprete denominado wish (para Tcl/Tk 8.0, wish80), que añade a los comandos de Tcl, comandos capaces de crean interfaces gráficos de usuario (Welch, 2003). Es decir, Tk permite crear ventanas, botones, menús, barras de scroll, y toda una serie de elementos, propios de la programación con ventanas. A todos estos elementos los denomina widgets.

Tk se distribuye junto con Tcl en un paquete denominado Tcl/Tk, que proporciona los dos intérpretes citados anteriormente. El intérprete de Tcl puede ser eliminado ya que se encuentra incorporado en el intérprete de Tk.

La gran evolución que ha sufrido el motor de simulación consistente en la combinación Tcl/Tk ha hecho que sean varios los desarrolladores de plataformas hardware que hayan comenzado a incluir intérpretes de estos lenguajes en sus entornos de simulación. Un calo ejemplo de esto lo constituye el paquete software ModelSim. 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.

Como se verá a lo largo de este trabajo, gracias a esta posibilidad de comunicación con el motor de simulación Tcl/Tk, en lugar de llevar a cabo las verificaciones de funcionalidad clásica en la que el usuario analiza formas de onda asociadas a diferentes señales, es posible desarrollar interfaces visuales en las que en tiempo de simulación hardware haya una actualización de los elementos visualizados y así potenciar la creación de entornos de simulación mucho más amigables y vistosos.

No obstante, esto no quita la posibilidad de que también sea posible llevar a cabo una simulación mediante el análisis de forma de ondas clásico. A modo de ejemplo, en 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*'.

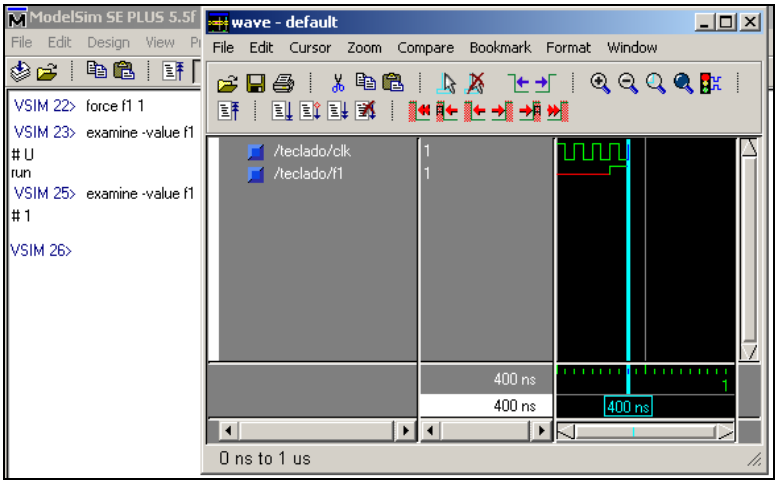


Fig. 1.1 Consola de ModelSim con visualizador de formas de onda.

2. Ejemplo de diseño basado en Tcl/Tk. Problema de la ruta óptima

A fin de mostrar la potencia de desarrollo de interfaces de simulación visuales, se va a aplicar a un ejemplo clásico de diseño con dispositivos FPGAs, consistente en un problema de determinación de la ruta óptima entre dos ciudades (Van den Bout, 1999). Se trata de generar un circuito combinacional que permita resolver la siguiente problemática:

El dueño de una empresa de ventas a domicilio detecta un gasto excesivo en concepto de vales de combustible y ticket de comidas, un análisis con profundidad del problema determina que el gasto es debido a una mala elección de las rutas por parte de sus vendedores. Las ciudades que cubre la empresa se muestran en la fig. 2.1

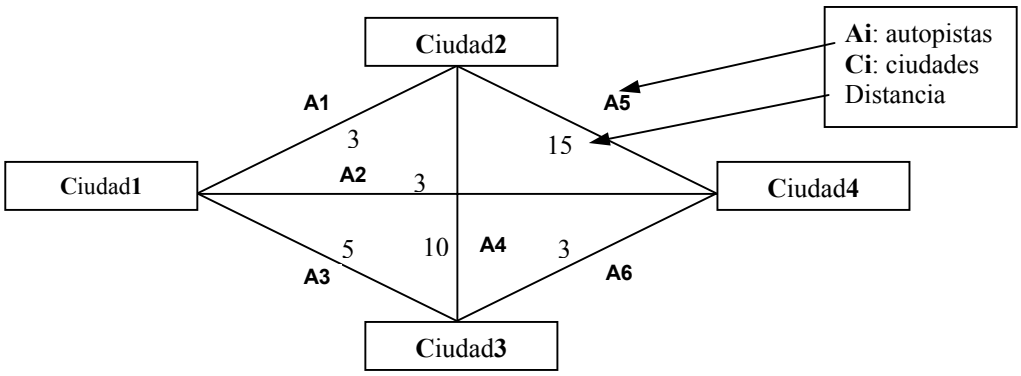


Figura 2.1. Esquema de rutas óptimas.

Se trata de diseñar un sistema inteligente (figura 2.2) que muestre a sus comerciales las rutas óptimas entre dos ciudades. El dispositivo constará de cuatro interruptores C1, C2, C3, C4 que corresponderán con las ciudades visitadas. Los interruptores pulsados dos a dos indicarán mediante un juego de LEDs la ruta óptima (A1,.., A6).

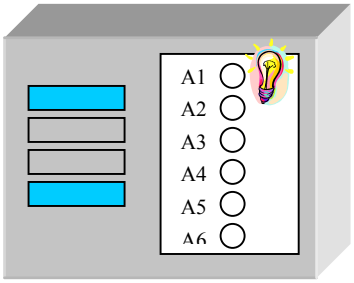


Figura 2.2. Sistema inteligente para determinación de la ruta óptima.

Tabla de verdad

Siguiendo el flujo de diseño habitual, el primer paso consiste en extraer la tabla de verdad correspondiente al problema propuesto, donde Ci (i=1,..4) corresponde a las diferentes ciudades y Ai corresponde con las autopistas (i=1,..6). Se trata de un sistema que al tener 4 entradas presentará una tabla de verdad que presente 16 formas posibles de codificación de las entradas. Sin embargo, realmente sólo se considerarán entradas válidas aquéllas en las que sólo haya dos ciudades seleccionadas, quedando estas situaciones reflejadas en la tabla 2.1. A fin de optimizar la minimización mediante mapas de Karnaugh, a las entradas no válidas se les otorga un valor indiferente.

Entradas				Salidas					
C1	C2	C3	C4	A1	A2	A3	A4	A5	A6
1	1	0	0	1	0	0	0	0	0
1	0	1	0	0	0	1	0	0	0
1	0	0	1	0	1	0	0	0	0
0	1	1	0	1	0	1	0	0	0
0	1	0	1	1	1	0	0	0	0
0	0	1	1	0	0	0	0	0	1

Tabla 2.1. Tabla de verdad

Ec. Booleanas:

Haciendo una minimización mediante mapas de Karnaugh considerando términos indiferentes, tal y como se ha comentado anteriormente, las ecuaciones booleanas correspondientes a la tabla de verdad se muestran a continuación.

$$A1 = C1 \cdot C2 \cdot \overline{C3} \cdot \overline{C4} + \overline{C1} \cdot C2 \cdot C3 \cdot \overline{C4} + \overline{C1} \cdot C2 \cdot \overline{C3} \cdot C4$$
$$A2 = C1 \cdot \overline{C2} \cdot \overline{C3} \cdot C4 + \overline{C1} \cdot C2 \cdot \overline{C3} \cdot C4$$
$$A3 = C1 \cdot \overline{C2} \cdot C3 \cdot \overline{C4} + \overline{C1} \cdot C2 \cdot C3 \cdot \overline{C4}$$
$$A4 = 0$$
$$A5 = 0$$
$$A6 = \overline{C1} \cdot \overline{C2} \cdot C3 \cdot C4$$

y agrupando términos quedarán:

$$A1 = C2 \cdot ((C1 \oplus C3) \cdot \overline{C4} + \overline{C1} \cdot \overline{C3} \cdot \overline{C4})$$
$$A2 = (C1 \oplus C2) \cdot \overline{C3} \cdot C4$$

$$A3 = (C1 \oplus C2) \cdot C3 \cdot \overline{C4}$$
$$A4 = 0$$
$$A5 = 0$$
$$A6 = \overline{C1} \cdot \overline{C2} \cdot C3 \cdot C4$$

Por ejemplo, el circuito lógico correspondiente con la salida A3 se muestra en la figura 2.3.

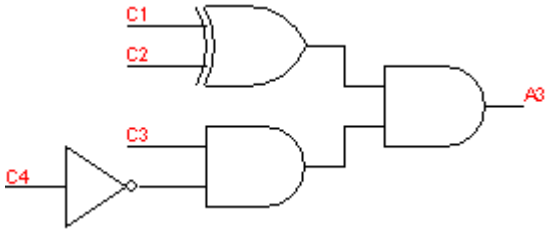


Figura 2.3. Circuito lógico. Salida A3.

Con estos datos, ya se dispone de las expresiones lógicas correspondientes a las salidas del sistema inteligente, por lo que es posible utilizar el software Xilinx ISE 6 para implementar la aplicación en lenguaje de descripción hardware (HDL), que en el trabajo aquí presentado se llevará a cabo utilizando VHDL.

3. Diseño lógico basado en VHDL

En este aparatado se va a analizar la metodología de diseño centrada en HDLs que utilizan las herramientas EDA actuales. En concreto, particularizaremos nuestra exposición para el software ISE 6 de la compañía Xilinx (Xilinx, 2005).

3.1 Código de la aplicación

Se ha realizado una descripción VHDL del sistema a nivel de transferencia de registros, especificando las distintas señales y el flujo de datos entre ellas. El código insertado corresponde con las ecuaciones *booleanas* extraídas de la tabla de verdad (tabla 2.1) para todas las salidas.

En este caso, debemos resaltar que las sentencias de asignación de señales (que utilizan el conectivo "<="), se comportan en realidad como procesos que tuvieran en su lista sensible todas las señales que aparezcan en la expresión de la parte derecha de la

secuencia de asignación, ejecutándose al igual que ellos de forma concurrente. El código VHDL correspondiente a tal descripción es el indicado a continuación:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Ruta_optima is
	port (
		C1, C2, C3, C4: in std_logic;
		A1, A2, A3, A4, A5, A6: out std_logic
	);
end Ruta_optima;

architecture Behavioral of Ruta_optima is
begin
	A1 <= C2 and (((C1 xor C3) and (not C4)) or ((not C1)and (not C3) and C4));
	A2 <= ((C1 xor C2) and (not C3) and C4);
	A3 <= ((C1 and (not C2)) or ((not C1) and C2)) and (C3 and (not C4));
	A4 <= '0';
	A5 <= '0';
	A6 <= (not C1) and (not C2) and C3 and C4;
end Behavioral;
```

3.2 Simulación del diseño

El siguiente paso consiste en la realización de la simulación funcional del diseño para verificar el correcto funcionamiento del mismo. Para ello se ha utilizado el software ModelSim SE 5.7d , el cual, como ya se ha comentado en un apartado anterior, permite observar el valor de las señales de entrada y salida de forma gráfica. Las señales de entrada y salida se observan en la figura 3.1.

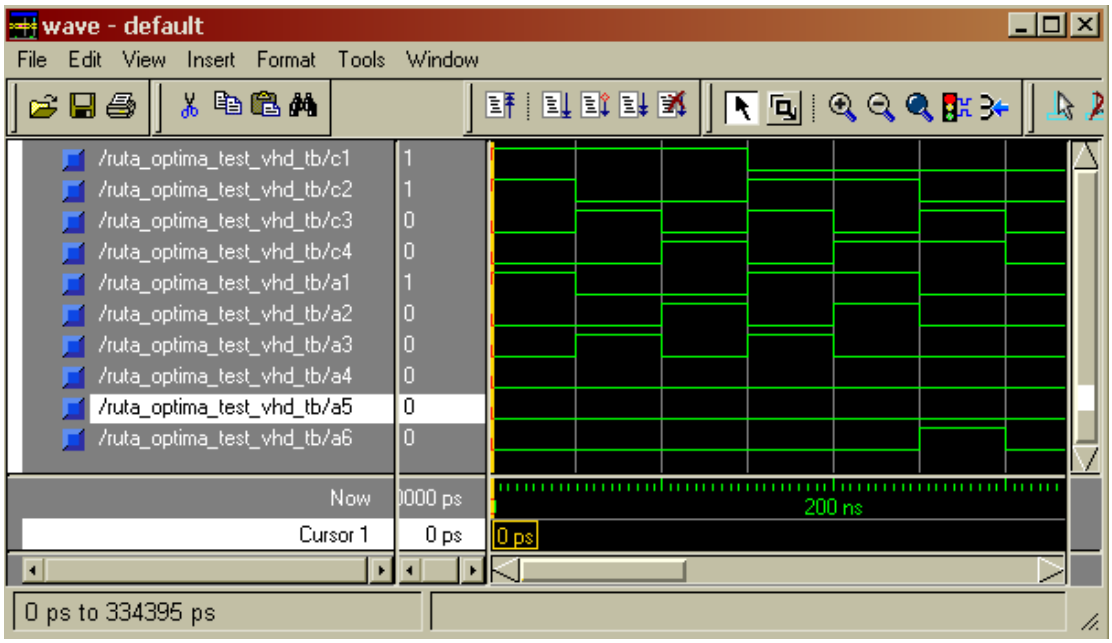


Figura 3.1. Simulación funcional.

3.3 Simulación mediante una interfaz gráfica implementada en Tcl/Tk

En este apartado se estudiará la programación en Tcl/Tk en el programa ModelSim y la utilización de programas en Tcl/Tk para interactuar con las simulaciones de ModelSim. La principal acción del programa ModelSim es la de realizar la simulación de diseños en VHDL. A continuación se van a enumerar algunos de los comandos que escritos en la consola de Modelsim permiten controlar los parámetros de la simulación, como son frecuencia de reloj, valor de las señales de entrada, etc.

Estos comandos serán utilizados en nuestro programa en Tcl/Tk y de esta forma se podrá actuar sobre la simulación directamente desde la ventana del programa en Tcl/Tk, para lo que se ha optado por una interfaz basada en botones. Los comandos de ModelSim se utilizarán primero para comenzar la simulación, y después para cambiar los valores de las entradas y para observar los valores de las salidas.

El comando *vlib* será escrito en la consola de *ModelSim* para la creación de una librería, necesaria para la posterior compilación del programa en *VHDL*. 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*. Una vez que se encuentra el programa compilado se procederá a simularlo con el comando *vsim,* escribiendo simplemente en la consola de *Modelsim* la palabra *vsim* y dentro del menú que aparece se seleccionará el archivo que se desea simular.

Una vez que se encuentra iniciada 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'* permite al usuario variar el estado de una señal en ModelSim a continuación se adjunta un ejemplo de utilización de este comando: *force nombre_señal 1* de esta forma con este código se consigue establecer el valor de la señal *nombre_señal* a 1.

- La instrucción *'examine'* es la utilizada para la comprobación del valor de una señal. En este proyecto se utilizará este comando asociado a un bucle *if* del siguiente modo: *if {[examine -value / nombre_señal] == 1}* { *instrucciones* } de esta forma el programa realiza las instrucciones contenidas en el bucle *if* siempre que se cumpla que la señal *nombre_señal* sea igual a 1.

3.3.1. Interfaz de simulación basada en Tcl/Tk

Se ha desarrollado una interfaz visual basada en el lenguje Tcl/Tk, para lo que se ha creado un 'frame', denominado principal, sobre el que han ido colocando el resto de componentes. A continuación se ha colocado sobre el anterior marco (frame) uno nuevo para albergar una serie de botones, y también se ha introducido un '*canvas*' para dibujar los elementos que representarán los LEDs. En el '*canvas*' se crean seis círculos que representarán los LEDs que se deben encender para escoger la ruta óptima (LEDs asociados a las salidas que representan el camino a tomar). A continuación se han situado en la GUI 4 botones de selección que pulsados dos a dos identifican las ciudades de partida y llegada, los 'checkbutton' tienen dos propiedades importantes: "-command" que salta a un procedimiento denominado *camino* cuando el botón es seleccionado; y, "-variable" asociada a una variable (ciu1, ciu2, ciu3, ciu4) que se pone a uno cuando se selecciona y a cero en caso contrario. Para una mejor comprensión del programa se han puesto varios widgets de texto.

Cuando se pulsa cualquier botón el programa en *tcl/tk* se comunica directamente con la simulación y hace variar el valor de las señales de salida A1-A6, cambio que se plasma en la actualización de los LEDs representando el camino óptimo, tal y como se puede apreciar en la figura 3.2.

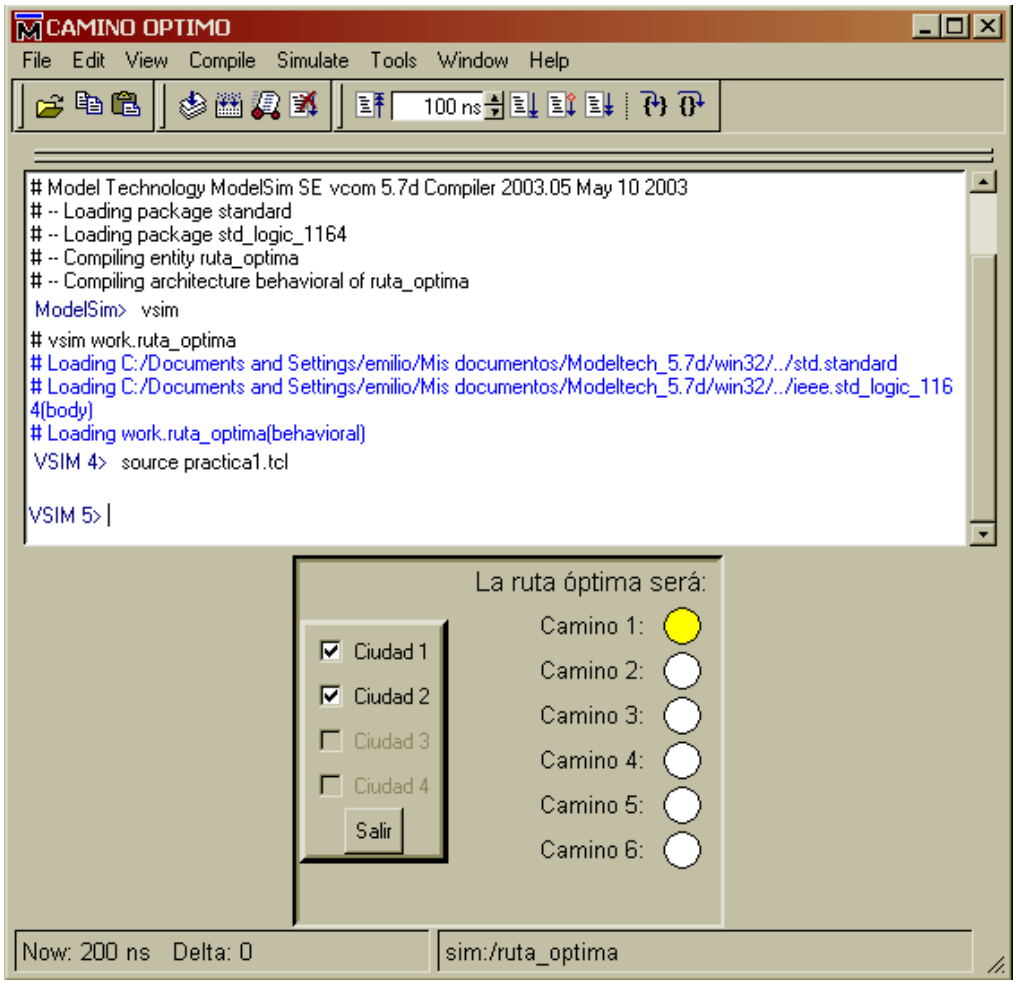


Figura 3.2. Aspecto de la interfaz gráfica y su ejecución.

Es conveniente para comprobar el correcto funcionamiento de la simulación utilizar el visualizador de señales que incorpora ModelSim, como apreciarse en la figura 3.3.

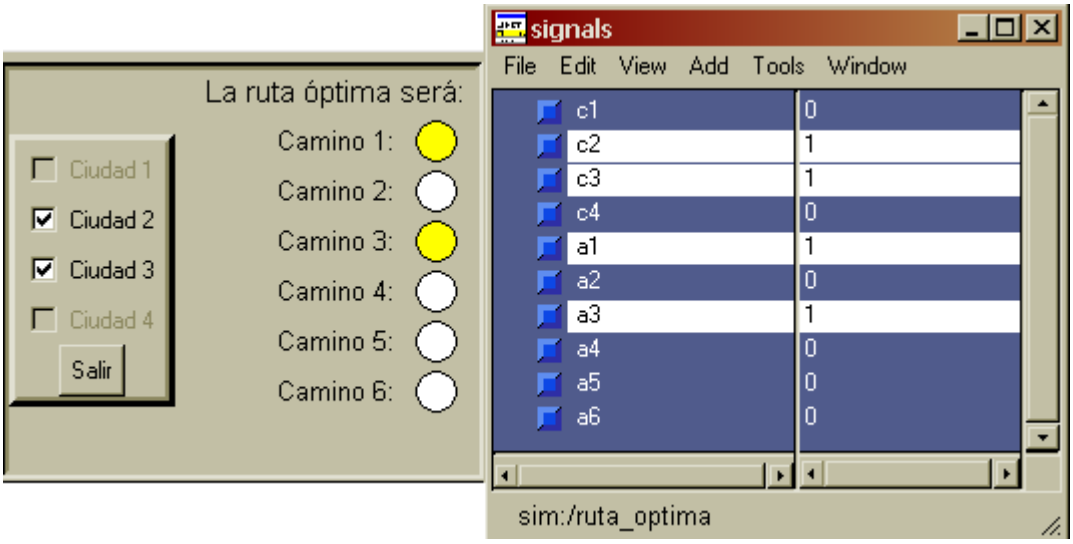


Figura 3.3. Funcionamiento del ModelSim junto con el visualizador de señales.

3.4. Simulación temporal

La simulación funcional del diseño no incluye tiempos de retardo entre rutas internas y la lógica del dispositivo reconfigurable. Estos retardos pueden generar errores en la transferencia de señales de reloj cuando e trabaja a frecuencias elevadas. Para verificar el funcionamiento real del diseño se acude a una simulación temporal permitida en el entorno de simulación en la que estos efectos se contemplan. Para ello es necesario una secuencia de procesos que se resume en la siguiente lista:

- Translate: el proceso de traducción todas las netlist de entrada en un formato compatible con el dispositivo seleccionado.
- Map: mapea un diseño lógico a una FPGA.
- Place & Route: Asigna las puertas lógicas del circuito a los CLBs y establece las rutas en el interior de la FPGA.
- FPGA editor: permite ver y modificar la implementación física, incluyendo el enrutado.
- Timing Analyzer: Calcula los retrasos en los CLBs y las rutas y los almacena en un archivo para ser utilizado en la simulación temporal.
- Chip Viwer: esta herramienta provee una visión gráfica de las entradas y salidas, detalles de las macroceldas, y la asignación de pines.

Una vez realizada la implementación se crean diferentes informes en cada uno de los pasos siendo los más importantes. El resultado que se obtiene de la simulación temporal viene representado en la figura 3.4.

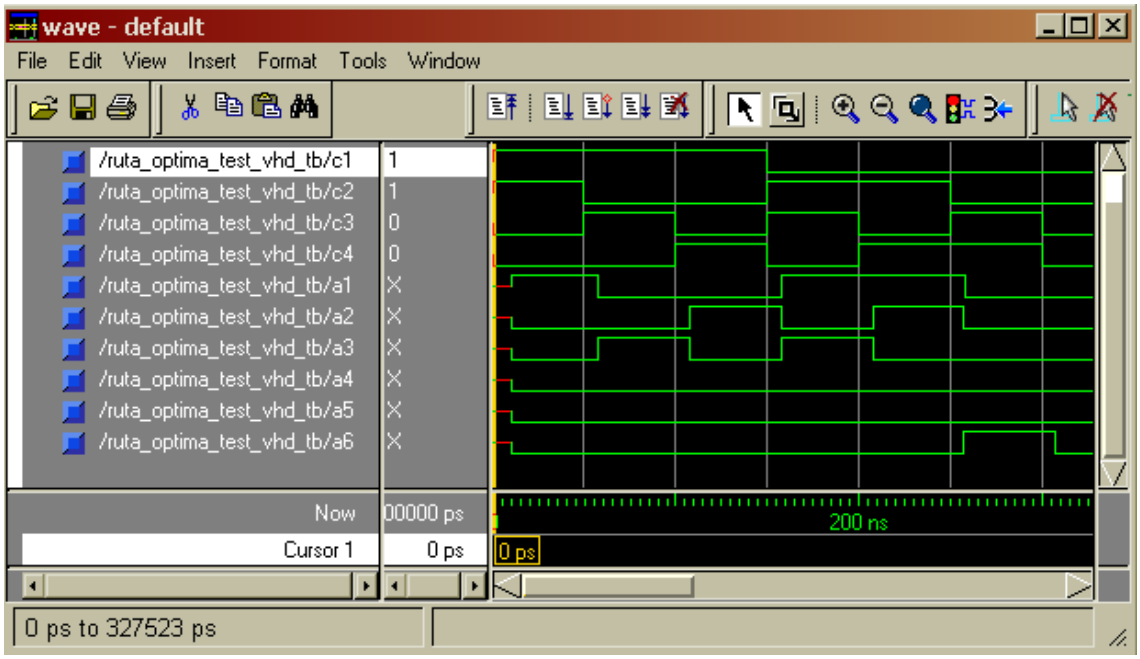


Figura 3.4. Simulación temporal.

Para poder observar el retardo nos fijamos en el punto en el que C2 cambia a 1, C1 cambia a 0 y C3 pasa a 1. En la figura 18 se aprecia que las señales que deben cambiar de estado como son A1, A2 y A3 no lo hacen inmediatamente sino transcurrido un tiempo $t \approx 8000$ ps = 8 ns.

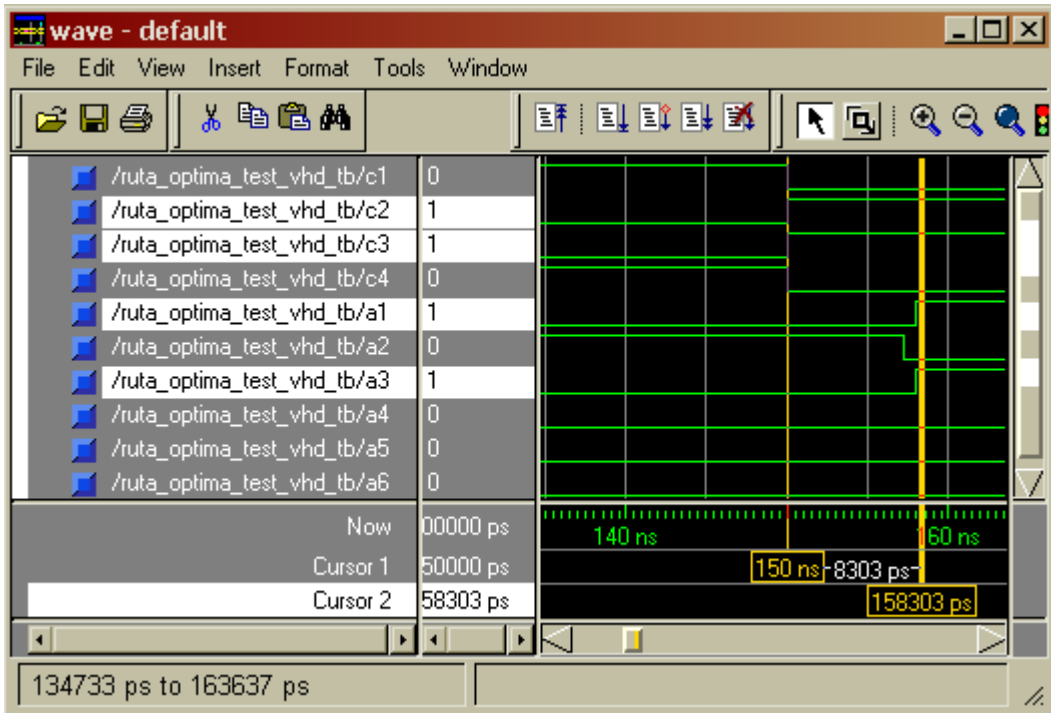


Figura 3.5. Detalle del retardo de propagación.

4. Conclusión

En este artículo se ha diseñado un sistema de búsqueda de una ruta óptima que permite a los dueños de las empresas de ventas a domicilio la elección la ruta más corta para sus vendedores con el fin de reducir los gastos excesivos en concepto de vales de combustible y ticket de comidas.

Se ha presentado las ventajas obtenidas por el uso de una interfaz visual, basada en el lenguaje Tcl/Tk, la cual permita una eficaz, fácil y sencilla verificación de la funcionalidad de núcleos hardware implementados con el lenguaje de descripción hardware VHDL y los dispositivos reconfigurables (FPGA), evitando así la tediosa forma de validación asociada al análisis de cronogramas en entornos clásicos de simulación.

5. Referencias

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

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

OUSTERHOUT, (2005): http://www.sunlabs.com:80/people/john.ousterhout

OUSTERHOUT (1994): John K. Ousterhout. "*Tcl and the Tk Toolkit*". Addison-Wesley Publishing Company, Inc. ISBN 0-201-63337-X

SUNLABS (2005): http://www.sunlabs.com/research/tcl

VAN DEN BOUT (1999): David Van de Bout. "*The practical Xilinx Designer's Lab Book*". Prentice Hall, 1999.

WELCH (2003): Brent Welch . "*Practical Programming in Tcl and Tk*". Prentice Hall. ISBN 0-13-182007-9

XILINX (2005): Xilinx Inc. http://www.xilinx.com