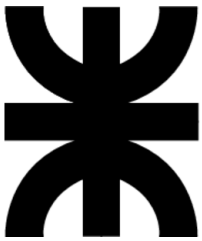
Universidad Tecnológica Nacional

Facultad Regional Córdoba



Carrera de Ingeniería Electrónica

"Técnicas Digitales I"

Curso de Tercer año - Anual

Tema:

Trabajo Final:
Diseño de controlador de trenes para transporte de Carbón

Profesores:

Titular: Ing. Toledo, Luis Eduardo
Adjunto: Ing. Gutierrez, Francisco Guillermo
JTP: Ing. Casasnovas, Marcelo Oscar

Alumnos:

Manolucos, Enzo Nicolas (legajo Nº: 74868)
Villegas, Sebastián (legajo Nº: 73643)

Índice

Introducción

En este trabajo, buscaremos poner en práctica los conocimientos adquiridos en la catedra de Técnicas Digitales I, desarrollando un sistema de control digital para dos trenes eléctricos. Nos planteamos diseñar un circuito de control automático para evitar colisiones y la circulación lo más segura posible de dichos trenes.

De esta manera tendremos un seguimiento de los trenes de manera segura. Así se obtiene información decisiva para una administración efectiva y sin dificultades. El registro de trenes es complejo y costoso, ya que muchos sistemas e instalaciones deben encadenarse y comunicarse entre ellos de manera perfecta.

Para lograr este cometido, haremos una descripción de hardware en Verilog de una máquina de estado finita (finite state machine), la cual implementaremos en una placa Basys 2, disponible en el laboratorio de técnicas digitales, que cuenta con una FPGA Spartan-3E de la marca Xilinx.

Objetivos

- Diseñar una máquina de estado capaz de lograr la correcta y segura operación de dos trenes eléctricos
- Realizar una descripción de hardware en Verilog para la implementación de esta máquina de estado
- Afianzar los conocimientos adquiridos de lógica secuencial

Desarrollo

Problema del control del tren

Tenemos dos minas de carbón, una principal y otra secundaria. También está la estación principal donde se llevan a cabo todos los controles, un puerto para enviar el carbón por barco y por ultima el deposito de carbón.

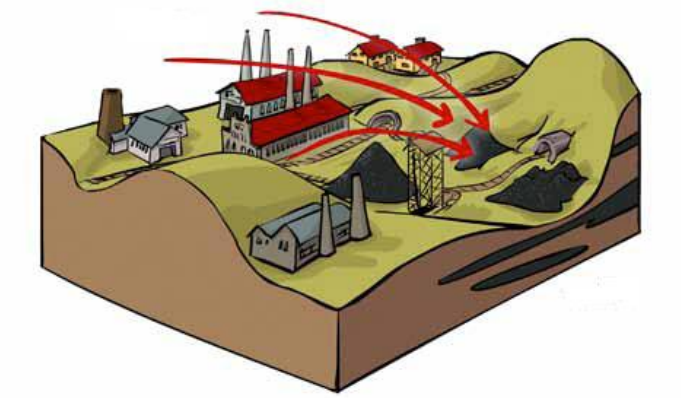


Figura 1 *Mina de carbón ilustrativa*

Para el trasporte de carbón y de trabajadores tenemos dos trenes, A para el carbón y B para los trabajadores. Estos deben correr por vías sin que se produzcan colisiones. Para evitar esta colisión los trenes requieren de un controlador de seguridad, que permita que los trenes se muevan dentro y fuera de las intersecciones sin que se produzca ningún contratiempo.

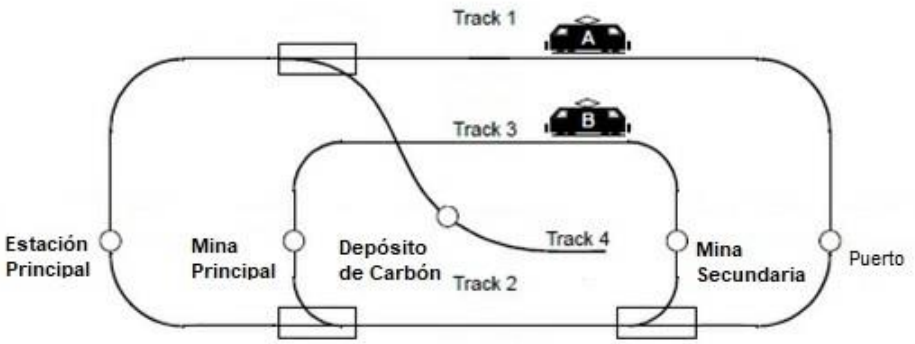


Figura 2 *Vías de los trenes*

La disposición de la vía en la figura 2 está dividida en cuatro segmentos, cada uno correspondiente a cada zona de la mina, donde cada tren tendrá su recorrido de acuerdo a su función. Para una operación segura, solo un tren a la vez puede estar presente sobre un determinado segmento de vía. Esto se logrará al crear una maquina de estado finita (fsm) que cambiará los carriles y el movimiento de los trenes de acuerdo a donde estén ubicados estos.

Salidas de direcciones de trenes

La dirección de cada tren es controlada por cuatro señales (dos por cada tren) TA0-TA1 para el tren A, y DB0-DB1. Cuando estas señales indican “01” hacia adelante para un tren en particular, un tren se moverá en sentido horario (sobre la vía 4, el tren se mueve hacia el bucle exterior). Cuando la señal implica “10” reversa, los trenes se moverán en sentido anti horario. El valor “11” es un valor no permitido y no debe ser usado. Cuando las señales son establecidas a “00”, el tren se detendrá. Como se observa en la figura 3.

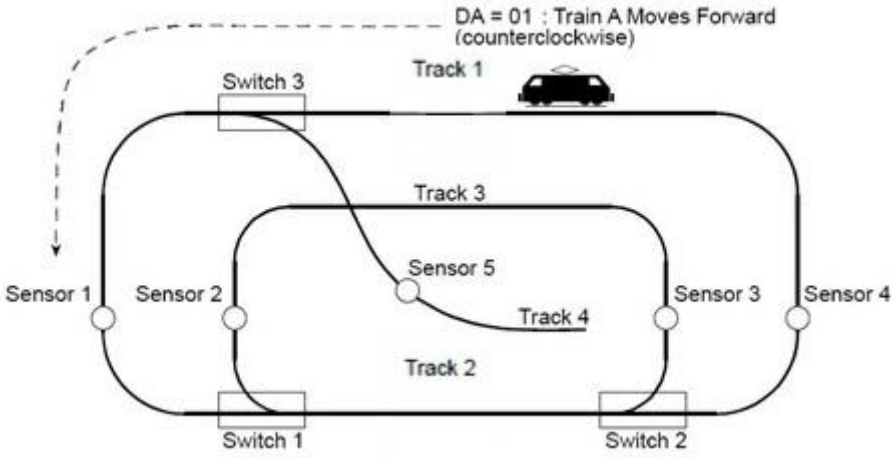


Figura 3 *Dirección de los trenes*

Salidas de switches de dirección

Los switches de dirección son controlados actuando sobre las señales de salidas SW1, SW2 y SW3, ya sea en alto (salida conectada con la vía interior) o en bajo (vías exteriores conectadas). Esto significa que en cualquier momento en que todos los switches son puestos a “1”, las vías están configuradas de manera tal que el bucle exterior está conectado al bucle interior como se en la figura 4.

Si un tren se mueve en dirección incorrecta a través de un switch abierto, el mismo se descarrilará. Si un tren se encuentra en el punto track 1 en la figura 4, y se está moviendo hacia la izquierda, se descarrilará en el switch 3. Para evitar descarrilamiento, SW3 debe ser puesto a 0. Además, notar que los tracks 3 y 4 se cruzan en una intersección por lo que se debe tener cuidado y evitar colisión en este punto.

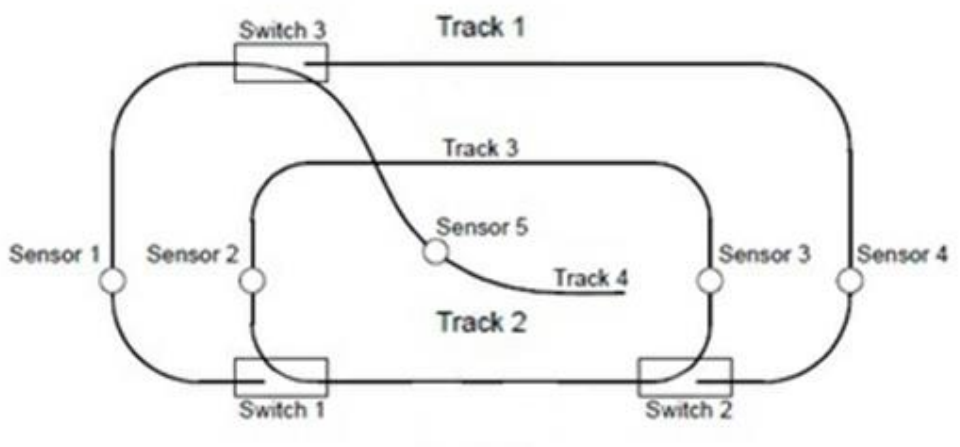


Figura 4 *Disposición de los switches*

Señales de entrada de sensores de los trenes

Las seis entradas de señales de sensores de los trenes S1, S2, S3, S4 y S5, cada una correspondiente a cada zona de la mina, se ponen en estado alto cuando un tren está cercano al sensor. Los sensores se disparan continuamente durante muchos ciclos de reloj por pasaje de cada tren. Las señales de entradas y salidas de la máquina de estado se ven en la figura 5.

Figura 5 *Señales de entrada y salida*

Entradas

- Sensor (S1, S2, S3, S4, S5): Tren Presente = 1, Tren Ausente = 0
- Recolección de Carbón (G): Carga = 1, Descarga = 0
- Dirección (D): Horario = 1, Antihorario = 0
- Inicio (P): Marcha = 1, Freno = 0
- Reset

Salidas

- Switches (SW1, SW2, SW3): Conectado a la vía exterior = 0, Conectado a la vía interior = 1
- Direccion de los trenes Trenes (TA0, TA1, TB0, TB1): Detenido = 00, Horario = 01, Antihorario 10
- Display 7 Segmentos: Indican el sensor sobre el cual el tren esta detenido o si se encuentra en movimiento.

Se utilizará una simulación de tren “virtual” basada en FPGA, que emula esta configuración y proporciona salida que se visualizará en leds de la placa FPGA como se ve en la figura 6, se busca dar una indicación visual de cómo las salidas del tren trabajan en el sistema real

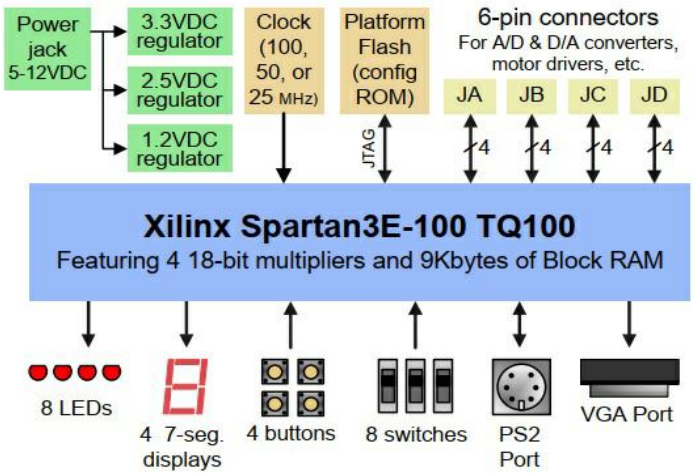


Figura 6 *Diagrama en bloques de la FPGA*

Funcionamiento

Tenemos tres señales que controlan el funcionamiento de los trenes, dirección D el cual funciona para cambiar el sentido de giro de ambos trenes, carga y descarga de carbón G el cual nos indica cuando el tren A va hacia el depósito de carbón y Inicio P con el cual iniciamos o paramos el movimiento de los trenes. Las salidas serán el cambio de los tres carriles mediante los switch y el sentido de giro de cada tren.

Tenemos dos displays los cuales muestran en caso de que un tren este detenido, el sensor sobre el cual están y en caso de que estén en movimiento lo indican con una letra “A” o “b” respectivamente para cada tren.

Debido a que estamos diseñando el control de trenes para una mina **ambos trenes siempre iniciaran y finalizaran sus recorridos en la estación y mina principal**. Suponemos también que el control de ambos trenes está ubicado en la estación principal, donde se accionaran las tres señales antes mencionadas. Con esto tenemos tres recorridos distintos:

Antihorario

Los trenes parten de la estación y mina principal cuando desde la estación principal se le indique la dirección (D=0) y que inicie el recorrido (P=1). En este recorrido se prioriza el movimiento del tren B ya que es el que trasporta a los trabajadores. Si se quieren frenar los trenes lo harán una vez llegados a los puntos de partida.

Horario

Los trenes parten de la estación y mina principal cuando desde la estación principal se le indique la dirección (D=1), que inicie el recorrido (P=1) y que además el tren A no deposite carbón (G=0). Para este recorrido se prioriza el movimiento del tren A ya que es el que trasporta el carbón hacia el puerto. Si se quieren frenar los trenes lo harán una vez llegados a los puntos de partida.

Carga y descarga de carbón

Los trenes parten de la estación y mina principal cuando desde la estación principal se le indique la dirección (D=1), que inicie el recorrido (P=1) y que además el tren A deposite carbón (G=1). Se prioriza el movimiento de del tren A hasta que llegue al depósito, luego el tren B se moverá por las vías internas mientras el tren A permanece quieto. Una vez que termine de cargar carbón (G=0), el tren A se moverá mientras el B permanece quieto pudiendo decidir si comenzar el recorrido de ambos trenes en sentido horario o antihorario.

Diseño de la FSM

Diagrama de estados

Para el diseño partimos de una máquina de moore para la cual la salida en un momento dado sólo depende de su estado en ese momento, mientras la transición al siguiente estado depende del estado en que se encuentre y de la entrada introducida. De esta manera obtenemos el siguiente diagrama de estado.

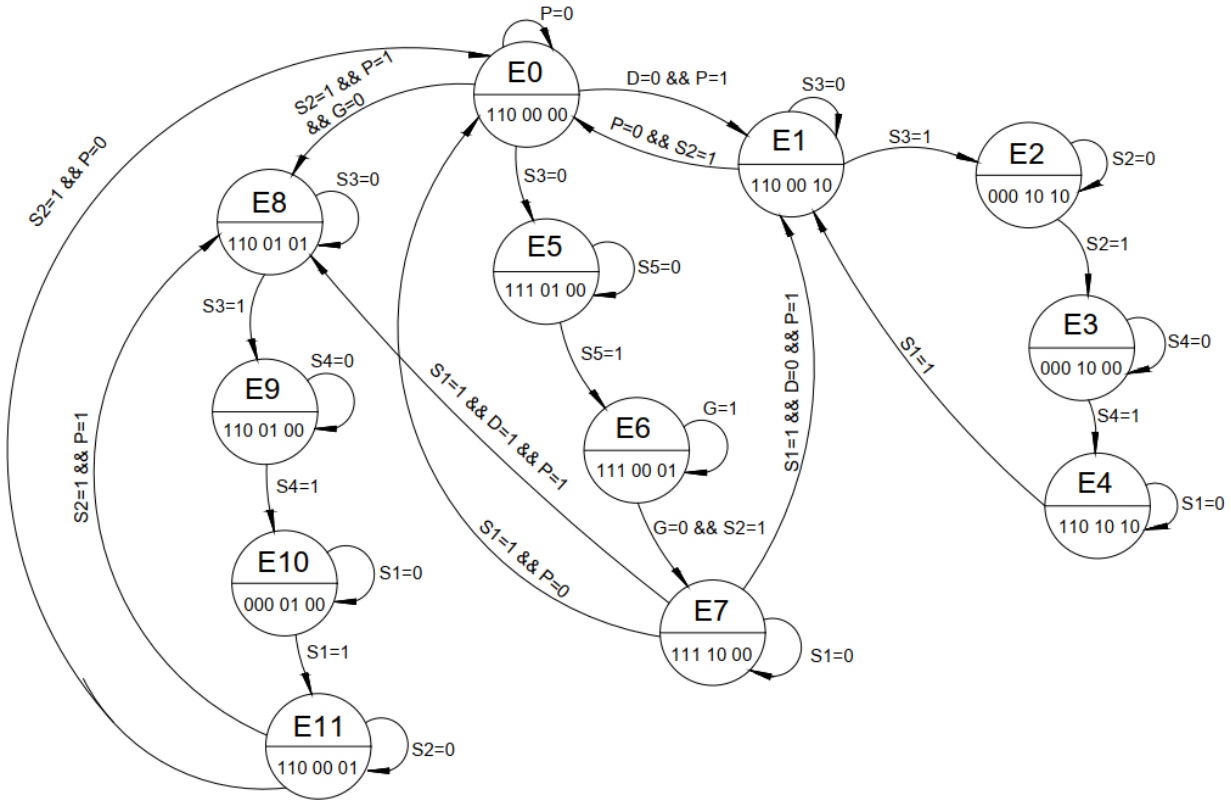


Figura 7 *Diagrama de estados*

Código Verilog

```
`timescale 1ns / 1ps

module fsm(
input clk, s1, s2, s3, s4, s5, reset, p, d, g, // p =1 avance, p=0 parado; d = 0 antihorario, d=1
horario;
output reg sw1, sw2, sw3, ta0, ta1, tb0, tb1,
output reg [3:0] anodos,
output reg [6:0] led);

parameter frecuencia = 1000000;
parameter freq_out = 5000;
parameter max_count = frecuencia/(2 * freq_out);

reg [3:0] estado;
reg [10:0] count;
reg clk_out;
reg [1:0] sel;

initial begin
        count = 0;
        clk_out = 0;
        sel = 0;
end
```

```
parameter e0=0, e1=1, e2=2, e3=3, e4=4, e5=5, e6=6, e7=7, e8=8, e9=9, e10=10, e11=11;

always @ (estado)
     begin
          case (estado)
               e0:
                  {sw1, sw2, sw3, ta0, ta1, tb0, tb1} = 7'b1100000;
               e1:
                  {sw1, sw2, sw3, ta0, ta1, tb0, tb1} = 7'b1100010;
               e2:
                  {sw1, sw2, sw3, ta0, ta1, tb0, tb1} = 7'b0001010;
               e3:
                  {sw1, sw2, sw3, ta0, ta1, tb0, tb1} = 7'b0001000;
               e4:
                  {sw1, sw2, sw3, ta0, ta1, tb0, tb1} = 7'b1101010;
               e5:
                  {sw1, sw2, sw3, ta0, ta1, tb0, tb1} = 7'b1110100;
               e6:
                  {sw1, sw2, sw3, ta0, ta1, tb0, tb1} = 7'b1110001;
               e7:
                  {sw1, sw2, sw3, ta0, ta1, tb0, tb1} = 7'b1111000;
               e8:
                  {sw1, sw2, sw3, ta0, ta1, tb0, tb1} = 7'b1100101;
               e9:
                  {sw1, sw2, sw3, ta0, ta1, tb0, tb1} = 7'b1100100;
               e10:
                  {sw1, sw2, sw3, ta0, ta1, tb0, tb1} = 7'b0000100;
               e11:
                  {sw1, sw2, sw3, ta0, ta1, tb0, tb1} = 7'b1100001;
               default:
                  {sw1, sw2, sw3, ta0, ta1, tb0, tb1} = 7'b1100000;
          endcase
     end

always @(posedge clk, negedge reset)
begin
       if (reset == 0)
              estado <= e0;
       else
       begin
              case (estado)
              e0:
                     if (d == 0 && p == 1) //Antihorario
                            estado <= e1;
                     else if (d == 1 && p == 1 && g == 0) //Horario
                            estado <= e8;
                     else if (d == 1 && p == 1 && g == 1) //Guardar
                            estado <= e5;
                     else
                            estado <= e0;
              e1:
                     if (s3 == 1 && p == 1)
                            estado <= e2;
                     else if (s2 == 1 && p == 1)
                            estado <= e3;
                     else if (s2 == 1 && p == 0)
                            estado <= e0;
                     else
                            estado <= e1;
              e2:
                     if (s2 == 1)
                            estado <= e3;
                     else
                            estado <= e2;
```

```
		e3:
			if (s4 == 1)
				estado <= e4;
			else
				estado <= e3;
		e4:
			if (s1 == 1)
				estado <= e1;
			else
				estado <= e4;
		e5:
			if (s5 == 1)
				estado <= e6;
			else
				estado <= e5;
		e6:
			if (g == 0 && s2 == 1)
				estado <= e7;
			else
				estado <= e6; // g == 1
		e7:
			if (s1 == 1 && d == 0 && p == 1)
				estado <= e1;
			else if (s1 == 1 && p == 0)
				estado <= e0;
			else if (s1 == 1 && d == 1 && p == 1)
				estado <= e8;
			else
				estado <= e7;
		e8:
			if (s3 == 1)
				estado <= e9;
			else
				estado <= e8;
		e9:
			if (s4 == 1)
				estado <= e10;
			else
				estado <= e9;
		e10:
			if (s1 == 1)
				estado <= e11;
			else
				estado <= e10;
		e11:
			if (s2 == 1 && p == 1)
				estado <= e8;
			else if (s2 == 1 && p == 0)
				estado <= e0;
			else
				estado <= e11;
		endcase
		end
end

always @ (posedge clk) //divisior de frecuencia
begin
	if (count == max_count)
		begin
			clk_out = ~clk_out;
			count = 0;
		end
		else
			count = count + 1;
```

```
end
always @ (posedge clk_out) //
begin
	if (sel == 1)
		sel <= 0;
	else
		sel <= sel + 1;
end
always @(posedge clk_out) //conmuta transistores
	case (sel)
		2'b00 : anodos <= 4'b1110;
		2'b01 : anodos <= 4'b1101;
	endcase
always @(*)
	begin
		case({estado, sel})
		6'b0000_00: led <= 7'b1001111; //trenA s1
		6'b0000_01: led <= 7'b0010010; //trenB s2
		6'b0001_00: led <= 7'b1001111; //trenA s1
		6'b0001_01: led <= 7'b1100000; //trenB mov
		6'b0010_00: led <= 7'b0001000; //trenA mov
		6'b0010_01: led <= 7'b1100000; //trenB mov
		6'b0011_00: led <= 7'b0001000; //trenA mov
		6'b0011_01: led <= 7'b0010010; //trenB s2
		6'b0100_00: led <= 7'b0001000; //trenA mov
		6'b0100_01: led <= 7'b1100000; //trenB mov
		6'b0101_00: led <= 7'b0001000; //trenA mov
		6'b0101_01: led <= 7'b0010010; //trenB s2
		6'b0110_00: led <= 7'b0100100; //trenA s5
		6'b0110_01: led <= 7'b1100000; //trenB mov
		6'b0111_00: led <= 7'b0001000; //trenA mov
		6'b0111_01: led <= 7'b0010010; //trenB s2
		6'b1000_00: led <= 7'b0001000; //trenA mov
		6'b1000_01: led <= 7'b1100000; //trenB mov
		6'b1001_00: led <= 7'b0001000; //trenA mov
		6'b1001_01: led <= 7'b0000110; //trenB s3
		6'b1010_00: led <= 7'b0001000; //trenA mov
		6'b1010_01: led <= 7'b0000110; //trenB s3
		6'b1011_00: led <= 7'b1001111; //trenA s1
		6'b1011_01: led <= 7'b1100000; //trenB mov
	endcase
	end
endmodule
```

Conclusión

La realización de este trabajo requirió tener en cuenta el funcionamiento de los trenes y diseñar la máquina de estado de la manera para que dicho funcionamiento sea lo más seguro posible para el problema que planteamos.

Implementamos una descripción de hardware en Verilog capaz de cubrir las necesidades que teníamos con este proyecto y para el desarrollo de dicha descripción, recurrimos a los apuntes de la catedra y a los trabajos prácticos realizados en el laboratorio.

Como conclusión final podemos destacar que realizamos un proyecto el cual tiene la capacidad de cambiar el sentido de circulación de dos trenes, permitiendo la operación segura de dichos trenes.

Bibliografía

- Thomas L. Floyd, *Fundamentos de sistemas digitales (9na edición),* Pearson, 2006.
- Stephen Brown, Zvonko Vranesic, *Fundamentos de Lógica Digital con Diseño VHDL (2da edición),* Mc Graw Hill, 2006.
- Joseph Cavanagh, *Verilog HDL Design Examples*, CRC Press, 2018.