Universidad de Costa Rica | Facultad de Ingeniería | Escuela de Ingeniería Eléctrica

Anteproyecto V

Roberto Sánchez Cárdenas — B77059
Gabriel Jiménez Amador — B73895

San José, 31 de agosto de 2021
IE0424 — Laboratorio de Circuitos Digitales

Índice

1. Introducción

Este laboratorio tiene como fin aprender a utilizar ciertos periféricos con la FPGA provista. En este caso se utiliza un mouse y teclado para controlar un juego de piedra papel o tijera.

2. Objetivos

- Investigar cómo utilizar periféricos en la tarjeta de desarrollo.
- Realizar interfaz entre un periférico y un procesador sencillo.

- Implementar un juego sencillo en la tarjeta de desarrollo.

3. Preguntas de investigación

3.1. ¿Qué señales forman parte de la interfaz PS2?

La interfaz PS2 posee 6 pines para transmisión de señales. [1]

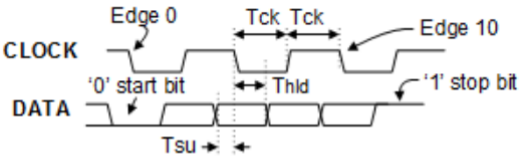
- Data
- Clock
- Tierra
- Vcc (+5V)
- Dos pines no implementados

3.2. ¿Cuál es el significado y la funcionalidad de cada una de las señales que conforman la interfaz?

- Data: Transmisión de datos generados por los sensores que poseen los periféricos conectados.
- Clock: Sincronización de señales
- Tierra: Puesta a tierra
- Vcc (+5V): Alimentación y referencia
- Dos pines no implementados

3.3. Documente cómo un mouse conectado a través del puerto USB de la tarjeta Nexys 4 DDR puede transmitir datos a través del protocolo PS2.

Tanto el mouse como un teclado utilizan palabras de 11 bits, donde se incluye un bit de inicio, un byte de datos (LSB primero), paridad impar y bit de parada. El diagrama de las señales de reloj y datos se ve de la siguiente manera: [2]



Symbol	Parameter	Min	Max
T_{CK}	Clock time	30us	50us
T_{SU}	Data-to-clock setup time	5us	25us
T_{HLD}	Clock-to-data hold time	5us	25us

Figura 1: Diagrama de tiempos

Las señales de datos y relojs se habilitan cuando hay transmisión, caso contrario se mantienen IDLE.

Cada vez que hay actividad en el mouse, se envían 3 palabras por medio del protocolo explicado previamente, por lo que se requieren 33 bits por cada lectura. [2]

3.4. Documente cómo interpretar los datos que se envían a través del protocolo PS2 para un mouse.

El mouse envía 3 palabras por cada lectura, las cuales son referentes a las posiciones relativas. Por ejemplo, si se mueve a la derecha se envía una coordenada positiva en X y a la izquierda se envía un parámetro negativo. Los valores de X y Y indica la tasa a la que se mueve el mouse, un valor más grande indica un movimiento mayor. [2]

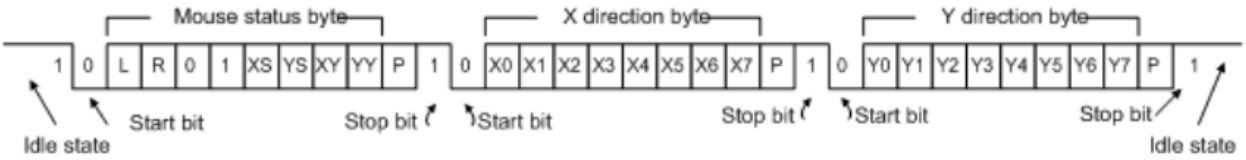


Figura 2: Envío de datos a través del protocolo PS2. [2]

El manual del Nexys 4 DDR, además menciona que el microcontrolador permite conocer el estado del *scroll wheel* con 4to byte de información para representar este otro "eje".

4. Diseño

Siguiendo los nombres y conexiones de los pines en el esquemático del Nexys 4 DDR en [3] se construye el archivo de restricción de pines en la Figura 3 y así incluir las conexiones del display de 7 segmentos y las del mouse con protocol PS/2.

```
set_property CFGBVS VCCO [current_design]
set_property CONFIG_VOLTAGE 3.3 [current_design]

# clk
set_property -dict { PACKAGE_PIN E3 IOSTANDARD LVCMOS33 } [get_ports { clk }];
create_clock -add -name sys_clk_pin -period 10.00 [get_ports {clk}];

# switches
set_property -dict { PACKAGE_PIN V10 IOSTANDARD LVCMOS33 } [get_ports { resetn }];

# leds
set_property -dict { PACKAGE_PIN H17 IOSTANDARD LVCMOS33 } [get_ports { out_byte[0] }];
set_property -dict { PACKAGE_PIN K15 IOSTANDARD LVCMOS33 } [get_ports { out_byte[1] }];
set_property -dict { PACKAGE_PIN J13 IOSTANDARD LVCMOS33 } [get_ports { out_byte[2] }];
set_property -dict { PACKAGE_PIN N14 IOSTANDARD LVCMOS33 } [get_ports { out_byte[3] }];
set_property -dict { PACKAGE_PIN R18 IOSTANDARD LVCMOS33 } [get_ports { out_byte[4] }];
set_property -dict { PACKAGE_PIN V17 IOSTANDARD LVCMOS33 } [get_ports { out_byte[5] }];
set_property -dict { PACKAGE_PIN U17 IOSTANDARD LVCMOS33 } [get_ports { out_byte[6] }];
set_property -dict { PACKAGE_PIN U16 IOSTANDARD LVCMOS33 } [get_ports { out_byte[7] }];

set_property -dict { PACKAGE_PIN V12 IOSTANDARD LVCMOS33 } [get_ports { out_byte_en }];
set_property -dict { PACKAGE_PIN V14 IOSTANDARD LVCMOS33 } [get_ports { trap }];

#7 segment display
# cathodes
set_property -dict { PACKAGE_PIN H15   IOSTANDARD LVCMOS33 } [get_ports { c_out[7] }];
set_property -dict { PACKAGE_PIN L18   IOSTANDARD LVCMOS33 } [get_ports { c_out[6] }];
set_property -dict { PACKAGE_PIN T11   IOSTANDARD LVCMOS33 } [get_ports { c_out[5] }];
set_property -dict { PACKAGE_PIN P15   IOSTANDARD LVCMOS33 } [get_ports { c_out[4] }];
set_property -dict { PACKAGE_PIN K13   IOSTANDARD LVCMOS33 } [get_ports { c_out[3] }];
set_property -dict { PACKAGE_PIN K16   IOSTANDARD LVCMOS33 } [get_ports { c_out[2] }];
set_property -dict { PACKAGE_PIN R10   IOSTANDARD LVCMOS33 } [get_ports { c_out[1] }];
set_property -dict { PACKAGE_PIN T10   IOSTANDARD LVCMOS33 } [get_ports { c_out[0] }];

# anodes
set_property -dict { PACKAGE_PIN U13   IOSTANDARD LVCMOS33 } [get_ports { an_out[7] }];
set_property -dict { PACKAGE_PIN K2    IOSTANDARD LVCMOS33 } [get_ports { an_out[6] }];
set_property -dict { PACKAGE_PIN T14   IOSTANDARD LVCMOS33 } [get_ports { an_out[5] }];
set_property -dict { PACKAGE_PIN P14   IOSTANDARD LVCMOS33 } [get_ports { an_out[4] }];
set_property -dict { PACKAGE_PIN J14   IOSTANDARD LVCMOS33 } [get_ports { an_out[3] }];
set_property -dict { PACKAGE_PIN T9    IOSTANDARD LVCMOS33 } [get_ports { an_out[2] }];
set_property -dict { PACKAGE_PIN J18   IOSTANDARD LVCMOS33 } [get_ports { an_out[1] }];
set_property -dict { PACKAGE_PIN J17   IOSTANDARD LVCMOS33 } [get_ports { an_out[0] }];

# mouse
set_property -dict { PACKAGE_PIN F4 IOSTANDARD LVCMOS33 } [get_ports { ps2_clk }];
set_property -dict { PACKAGE_PIN B2 IOSTANDARD LVCMOS33 } [get_ports { ps2_data }];
```

Figura 3: Archivo de restricciones de localización de pines a utilizar.

4.1. Ejercicio 1

Dado que se desea implementar un juego de piedra, papel o tijera, se debe tener tener la capacidad de mostrar ciertos mensajes en pantalla. La FPGA utilizada posee pantallas de 7 segmentos, por lo que se deben codificar los mensajes en ellas. Se creó un módulo que tiene estas capacidades y el código se presenta en la siguiente figura.

```
module display_control (
    input [3:0] selector,
    input clk,
    input reset,
output reg [7:0] c_out
    output reg [7:0] an_out,
);
reg [20:0] counter
always @(posedge clk) begin
    if(reset==0) counter<= 0;
    else counter <= counter+ 1;
end
always @(*) begin
case(selector):
    0: begin
        if (counter[20:18] == 0) begin //t
            c_out = 8'b10000111;
            an_out = 8'b11111110;
        end
        else if (counter[20:18] == 1) begin //r
            c_out = 8'b10101111;
            an_out = 8'b11111101;
        end
        else if (counter[20:18] == 2) begin //a
            c_out = 8'b10001000;
            an_out = 8'b11111011;
        end
        else if (counter[20:18] == 3) begin //t
            c_out = 8'b10000111;
            an_out = 8'b11110111;
        end
        else if (counter[20:18] == 4) begin //s
            c_out  = 8'b11010010;
            an_out = 8'b11101111;
        end
        else if (counter[20:18] == 5) begin
            c_out = 8'b11111111;
            an_out = 8'b11011111;
        end
        else if (counter[20:18] == 6) begin
            c_out = 8'b11111111;
            an_out = 8'b10111111;
        end
        else if (counter[20:18] == 7) begin
            c_out = 8'b11111111;
            an_out = 8'b01111111;
        end
    end
    1: begin
        if (counter[20:18] == 0) begin //t
            c_out = 8'b10000111;
            an_out = 8'b11111110;
        end
        else if (counter[20:18] == 1) begin //c
            c_out = 8'b11000110;
            an_out = 8'b11111101;
        end
        else if (counter[20:18] == 2) begin //e
            c_out = 8'b10000110;
            an_out = 8'b11111011;
        end
        else if (counter[20:18] == 3) begin //l
            c_out = 8'b11000111;
            an_out = 8'b11110111;
        end
        else if (counter[20:18] == 4) begin //e
            c_out = 8'b10000110;
            an_out = 8'b11101111;
        end
        else if (counter[20:18] == 5) begin //s
            c_out  = 8'b11010010;
            an_out = 8'b11011111;
        end
        else if (counter[20:18] == 6) begin
            c_out = 8'b11111111;
            an_out = 8'b10111111;
        end
        else if (counter[20:18] == 7) begin
            c_out = 8'b11111111;
            an_out = 8'b01111111;
        end
    end
```

```
    2: begin
        if (counter[20:18] == 0) begin //r
                c_out =  8'b10101111;
                an_out = 8'b11111110;
        end
        else if (counter[20:18] == 1) begin //e
                c_out = 8'b10000110;
                an_out = 8'b11111101;
        end
        else if (counter[20:18] == 2) begin //p
                c_out = 8'b10001100;
                an_out = 8'b11111011;
        end
        else if (counter[20:18] == 3) begin //a
                c_out = 8'b10001000;
                an_out = 8'b11110111;
        end
        else if (counter[20:18] == 4) begin //p
                c_out = 8'b10001100;
                an_out = 8'b11101111;
        end
            else if (counter[20:18] == 5) begin
                c_out = 8'b11111111;
                an_out = 8'b11011111;
        end
        else if (counter[20:18] == 6) begin
                c_out = 8'b11111111;
                an_out = 8'b10111111;
        end
        else if (counter[20:18] == 7) begin
                c_out = 8'b11111111;
                an_out = 8'b01111111;
        end
    end
    3: begin
        if (counter[20:18] == 0) begin //s
            c_out = 8'b11010010;
            an_out = 8'b11111110;
        end
        else if (counter[20:18] == 1) begin //r
            c_out =  8'b10101111;
            an_out = 8'b11111101;
        end
        else if (counter[20:18] == 2) begin //o
            c_out = 8'b10100011;
            an_out = 8'b11111011;
        end
        else if (counter[20:18] == 3) begin //s
            c_out = 8'b11010010;
            an_out = 8'b11110111;
        end
        else if (counter[20:18] == 4) begin //s
            c_out = 8'b11010010;
            an_out = 8'b11101111;
        end
        else if (counter[20:18] == 5) begin //i
            c_out = 8'b11001111;
            an_out = 8'b11011111;
        end
        else if (counter[20:18] == 6) begin //c
            c_out = 8'b11000110;
            an_out = 8'b10111111;
        end
        else if (counter[20:18] == 7) begin //s
            c_out = 8'b11010010;
            an_out = 8'b01111111;
        end
end
```

```
    4: begin
        if (counter[20:18] == 0) begin //k
            c_out = 8'b10001010;
            an_out = 8'b11111110;
        end
        else if (counter[20:18] == 1) begin //c
            c_out = 8'b11000110;
            an_out = 8'b11111101;
        end
        else if (counter[20:18] == 2) begin //o
            c_out = 8'b10100011;
            an_out = 8'b11111011;
        end
        else if (counter[20:18] == 3) begin //r
            c_out =  8'b10101111;
            an_out = 8'b11110111;
        end
        else if (counter[20:18] == 4) begin
            c_out = 8'b11111111;
            an_out = 8'b11101111;
        end
            else if (counter[20:18] == 5) begin
            c_out = 8'b11111111;
            an_out = 8'b11011111;
        end
        else if (counter[20:18] == 6) begin
            c_out = 8'b11111111;
            an_out = 8'b10111111;
        end
        else if (counter[20:18] == 7) begin
            c_out = 8'b11111111;
            an_out = 8'b01111111;
        end
end

    5: begin
        if (counter[20:18] == 0) begin //l
            c_out =  8'b11000111;
            an_out = 8'b11111110;
        end
        else if (counter[20:18] == 1) begin //a
            c_out = 8'b10001000;
            an_out = 8'b11111101;
        end
        else if (counter[20:18] == 2) begin //v
            c_out = 8'b11010101;
            an_out = 8'b11111011;
        end
        else if (counter[20:18] == 3) begin //i
            c_out = 8'b11001111;
            an_out = 8'b11110111;
        end
        else if (counter[20:18] == 4) begin //r
            c_out  = 8'b10101111;
            an_out = 8'b11101111;
        end
        else if (counter[20:18] == 5) begin
            c_out = 8'b11111111;
            an_out = 8'b11011111;
        end
        else if (counter[20:18] == 6) begin
            c_out = 8'b11111111;
            an_out = 8'b10111111;
        end
        else if (counter[20:18] == 7) begin
            c_out = 8'b11111111;
            an_out = 8'b01111111;
        end
    end
```

```
    6: begin
        if (counter[20:18] == 0) begin //n
            c_out = 8'b10101011;
            an_out = 8'b11111110;
        end
        else if (counter[20:18] == 1) begin //o
            c_out = 8'b10100011;
            an_out = 8'b11111101;
        end
        else if (counter[20:18] == 2) begin //w
            c_out = 8'b10010101;
            an_out = 8'b11111011;
        end
        else if (counter[20:18] == 3) begin //u
            c_out = 8'b1111111;
            an_out = 8'b1110111;
        end
        else if (counter[20:18] == 4) begin //u
            c_out = 8'b11100011;
            an_out = 8'b11101111;
        end
        else if (counter[20:18] == 5) begin //o
            c_out = 8'b10100011;
            an_out = 8'b11011111;
        end
        else if (counter[20:18] == 6) begin //y
            c_out = 8'b10010001;
            an_out = 8'b10111111;
        end
        else if (counter[20:18] == 7) begin
            c_out = 8'b11111111;
            an_out = 8'b01111111;
        end
    end


    7: begin
        if (counter[20:18] == 0) begin //t
            c_out =  8'b10000111;
            an_out = 8'b11111110;
        end
        else if (counter[20:18] == 1) begin //s
            c_out  = 8'b11010010;
            an_out = 8'b11111101;
        end
        else if (counter[20:18] == 2) begin //o
            c_out = 8'b10100011;
            an_out = 8'b11111011;
        end
        else if (counter[20:18] == 3) begin //l
            c_out = 8'b11000111;
            an_out = 8'b11110111;
        end
        else if (counter[20:18] == 4) begin
            c_out = 8'b11111111;
            an_out = 8'b11101111;
        end
        else if (counter[20:18] == 5) begin //u
            c_out =  8'b11100011;
            an_out =  8'b11011111;
        end
        else if (counter[20:18] == 6) begin //o
            c_out = 8'b10100011;
            an_out = 8'b10111111;
        end
        else if (counter[20:18] == 7) begin //y
            c_out = 8'b10010001;
            an_out = 8'b01111111;
        end
    end
```

```
    8: begin
         if (counter[20:18] == 0) begin //e
             c_out = 8'b10000110;
             an_out = 8'b11111110;
         end
         else if (counter[20:18] == 1) begin //i
             c_out = 8'b11001111;
             an_out = 8'b11111101;
         end
         else if (counter[20:18] == 2) begin //t
             c_out = 8'b10000111;
             an_out = 8'b11111011;
         end
         else if (counter[20:18] == 3) begin
             c_out = 8'b11111111;
             an_out = 8'b11110111;
         end
         else if (counter[20:18] == 4) begin
             c_out = 8'b11111111;
             an_out = 8'b11101111;
         end
         else if (counter[20:18] == 5) begin
             c_out = 8'b11111111;
             an_out = 8'b11011111;
         end
         else if (counter[20:18] == 6) begin
             c_out = 8'b11111111;
             an_out = 8'b10111111;
         end
         else if (counter[20:18] == 7) begin
             c_out = 8'b11111111;
             an_out = 8'b01111111;
         end
     end
end
endmodule
```

Figura 4: Módulo para desplegar texto deseado

Para la convención de las letras se utilizó la siguiente tabla. Cada uno de representa una letra.

Figura 5: Códigos para pantalla de 7 segmentos

4.2. Ejercicio 2

En el ejercicio 2 del laboratorio se debe implementar un juego de *Piedra, Papel o Tijeras* en el Nexys 4 DDR, utilizando un mouse con protocolo PS/2 y el display de 7 segmentos para poder jugar. El diagrama de estados diseñado para el juego se muestra en la Figura 6 donde en cada estado se realiza:

- START: Se muestra `START` en el display de 7 segmentos por 3 segundos.
- SELECT: Se muestra el texto `SELECT:` en el display de 7 segmentos por 1 segundo y luego pasa a mostrar `Paper`, si se mueve el scroll wheel para abajo, se muestra `Rock`, si se mueve para abajo de nuevo, se muestra `Scissors` (una vez más y muestra `Paper` y de forma similar si lo mueve para arriba). Según lo que muestra el display, cuando el usuario haga click derecho, esa será su elección. El eje del *scroll wheel* corresponde al 4to byte enviado por el mouse.
- SELECTED: Se espera 1 segundo luego de que el usuario haya realizado su elección.
- RIVAL: Utilizando un módulo de creación de números psuedo-aleatorios con LSFR, como el creado aquí, se elige algún valor entre 0, 1 y 2 que representarían piedra, papel y tijeras, respectivamente. Este valor es el que elige el contrincante máquina y es desplegado en el display de 7 segmentos.
- RESULT: Se compara los valores elegidos de la siguiente forma:
  - Usuario y CPU eligen igual: desplegar `TIE`
  - Según las elecciones se obtienen alguna de estas combinaciones:
    - Usuario: Piedra, CPU: Tijeras
    - Usuario: Papel, CPU: Piedra
    - Usuario: Tijeras, CPU: Papel

    Desplegar `You win`
  - Si no se cumple ninguna de las anteriores: desplegar `You lost`

  Cual sea el resultado, este se desplegará hasta que el usuario presione click derecho, en tal caso se regresa al estado inicial START.

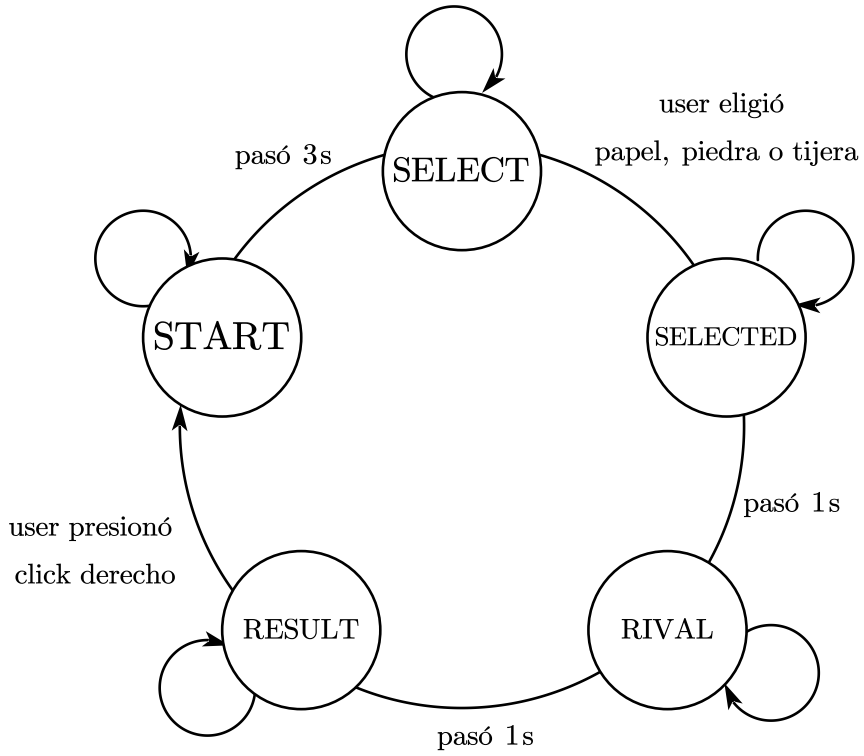


Figura 6: Diagrama de estados del juego de *Piedra, Papel o Tijeras*

Referencias

[1] Adam Chapweske. *PS/2 Mouse/Keyboard Protocol*. URL: http://www.burtonsys.com/ps2_chapweske.htm.

[2] Inc. Digilent. *Nexys 4 DDR Reference Manual*. 2016. URL: https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-4-ddr/nexys4ddr_rm.pdf.

[3] Inc. Digilent. *Nexys 4 DDR Schematic*. 2014. URL: https://reference.digilentinc.com/_media/reference/programmable-logic/nexys-4-ddr/nexys-4-ddr_sch.pdf.