INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

DISEÑO DE SISTEMAS DIGITALES

ENTREGABLE 1

Registro de Desplazamiento: Parallel-In Parallel-Out

Grupo: 4CV5

Integrantes:

Álvarez Hernández Gabriel Alexander

Bueno Aguilar Alexis Haziel

García Quiroz Gustavo Iván

Huesca Laureano Josué Alejandro

Muñoz Valdivia Irving Omar

Pedroza Villagómez Emir

PROFESOR: FLORES ESCOBAR JOSE ANTONIO

INTRODUCCION

Los registros son circuitos digitales con dos funciones básicas:

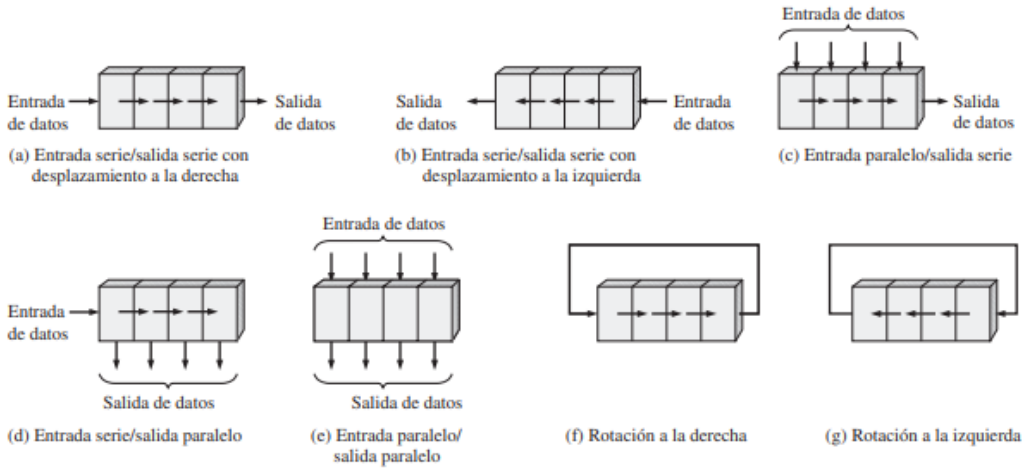
1. **Almacenamiento de datos**.

La capacidad de almacenamiento de datos es el número total de bits (1s y 0s) de un dato digital que puede contener. Esto es, cada flip-flop o etapa de un registro representa un bit de su capacidad de almacenamiento; por tanto, el numero de flip-flops en un registro determina la capacidad de almacenamiento de este.

2. **Desplazamiento de datos.**

Permite el movimiento de los datos de una etapa a otra dentro del registro, en función de los impulsos de reloj que se apliquen.

Los movimientos básicos de los datos en los registros de desplazamiento se ilustran en la siguiente figura.



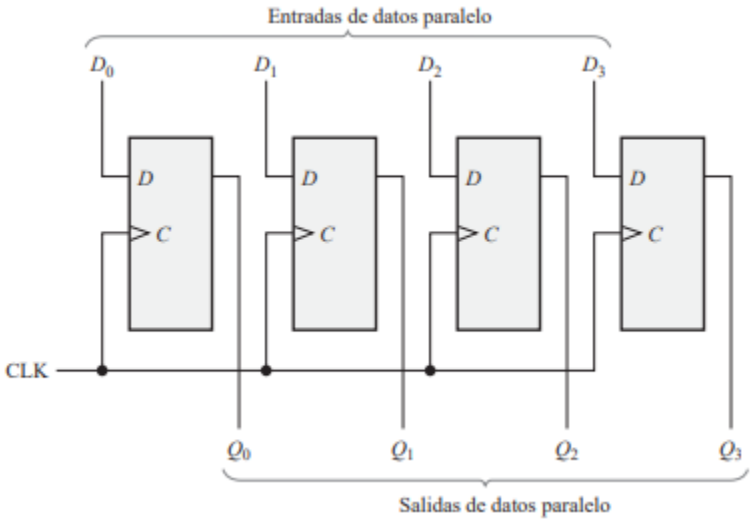
En este caso, nuestro enfoque principal se centrará en la entrada y salida en paralelo de los datos.

REGISTROS CON ENTRADA Y SALIDA EN PARALELO

En las entradas en paralelo, los bits se introducen simultáneamente en sus respectivos flip-flops a través de líneas paralelo, en lugar de bit a bit a través una única línea como ocurre con las entradas de datos serie.

En las salidas en paralelo, se dispone de la salida de cada flip-flop.

Así entonces, inmediatamente después de introducir simultáneamente todos los bits de datos, éstos aparecen en paralelo en las salidas paralelo.



Tanto las entradas como salidas son accesibles. Tienen un uso practico principal en los cálculos aritméticos

En esta sección de la práctica se realizará la codificación utilizando el lenguaje Verilog de la implementación de un **registro con entrada y salida en paralelo**, así como la visualización del RTL y la simulación de las diferentes señales involucradas.

DESARROLLO

CODIFICACION

Se crearon 4 archivos diferentes que conforman al registro de desplazamiento.

- FFD.v
- DivFreq.v
- FDD_Ctrl.v
- FDD_FPGA.v

FFD.v

Crea un módulo para un flip-flop tipo D, los comentarios dentro del código, al igual que en el resto, explican con más detalle el funcionamiento de este.

```
1   /* PROYECTO:       FDD_ParallelInParallelOut
2
3      ARCHIVO:        FDD.V
4      ASIGNATURA:     DSD
5      DESCRIPCION:    Descripción de un flip-flop tipo D
6      PROF:           Flores Escobar Jose Antonio
7      EQUIPO:         Álvarez Hernández Gabriel Alexander
8                      Bueno Aguilar Alexis Haziel
9                      Huesca Laureano Josue Alejandro
10                     García Quiroz Gustavo Ivan
11                     Muñoz Valdivia Irving Omar
12                     Pedroza Villagomez Emir
13  */
14
15  // Definicion de puertos
16  module FFD(
17     input d_i,           // Entrada de datos
18     input clk_i,         // Entrada de señal de reloj
19     input rst_i,         // Señal de reset para reestablecer el estado del FFD
20     output reg q_o       // Salida del dato almacenado en el FFD
21  );
22
23  // Definición de un flip-flop tipo D
24     always @(posedge clk_i, posedge rst_i)
25     // Siempre que haya un flanco positivo del clk hace esto...
26     begin
27        if(rst_i)         // El reset siempre tiene la más alta prioridad
28           q_o <= 1'b0;   // Si hay un reset el dato de salida es 0 (se reestablece)
29        else
30           q_o <= d_i;    // El dato de salida adopta el valor de entrada en caso contrario
31     end
32  endmodule
33
```

La salida **q_o** cambia en cada flanco de subida de reloj, a menos que exista una señal de **RESET**, que ocasionara que la salida se restablezca a un valor de 0.

DivFreq.v

Crea un módulo de un divisor de frecuencias.

Este será utilizado para crear una frecuencia reducida de una señal de reloj entrante cuya frecuencia es alta, dicha reducción esta dada por un factor de división (que en este caso será de 25,000,000).

```
1   /* PROYECTO:       FDD_ParallelInParallelOut
2
3      ARCHIVO:        DivFreq.v
4      ASIGNATURA:     DSD
5      DESCRIPCION:    Descripción de modulo que implementa un divisor de frecuencias
6      PROF:           Flores Escobar Jose Antonio
7      EQUIPO:         Álvarez Hernández Gabriel Alexander
8                      Bueno Aguilar Alexis Haziel
9                      Huesca Laureano Josue Alejandro
10                     Garcia Quiroz Gustavo Ivan
11                     Muñoz Valdivia Irving Omar
12                     Pedroza Villagomez Emir
13  */
14
15  // Definicion de puertos
16  module DivFreq(
17     input       clk_i,      // Entrada de señal de reloj
18     input       rst_i,      // Entrada de reset
19     output reg  clk_o       // Salida de la señal de reloj resultante
20  );
21
22  // Definición de señales
23     reg [31:0] ctr;         // Contador descendente de 32 bits
24
25  // Definicion de un divisor de frecuencias
26     always @(posedge clk_i, posedge rst_i)
27     // Siempre que haya un flanco positivo del clk hace esto...
28     begin
29        if(rst_i)            // Si existe una señal de reset...
30           begin
31              ctr <= 32'b0;  // Inicializamos el contador a 0
32              clk_o <= 1'b1; // La señal de reloj de salida se establece en 1
33           end
34        else                             // En caso contrario...
35           begin
36              if(ctr <= 32'd25 000 000)  // Si el contador llega al factor deseado...
37                 begin
38                    ctr <= 32'b0;        // El contador se reinicia
39                    clk_o <= ~clk_o;     // La señal de reloj se invierte
40                 end
41              else                       // En caso contrario...
42                 ctr <= ctr + 1'b1;      // Incrementa el contador
43           end
44     end
45  endmodule
```

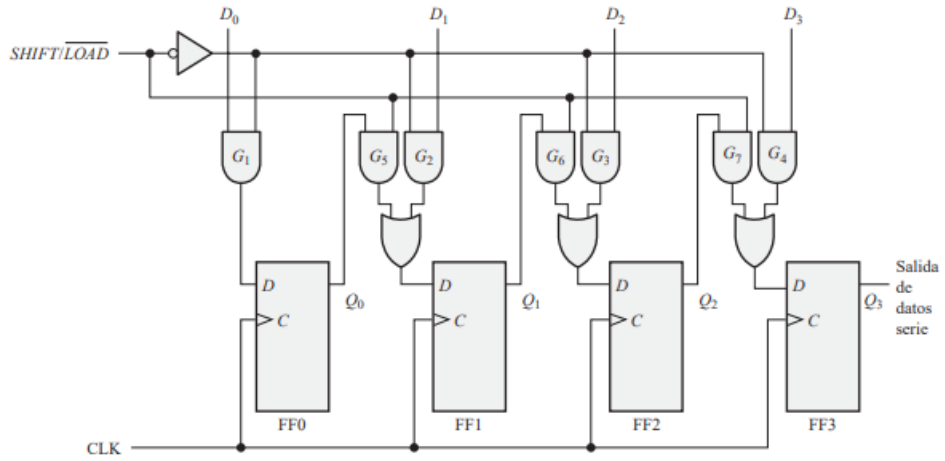
Como se indica en el código, si el contador llega al factor de división de 25,000,000 la señal de reloj se invierte a fin de reducir la frecuencia de esta, **que posteriormente será utilizada para el registro de desplazamiento.**

FDD_Ctrl.v

Crea un módulo de control para lo que serían los datos de entrada del registro, pues la entrada de estos se da en paralelo.

Se basa en mover o cargar los datos que se estén tratando, esto se maneja mediante una compuerta NOT.

El siguiente diagrama de un registro con entrada en paralelo y salida de 4 bits en serie describe su comportamiento. **Ya que aquí también estamos tratando entradas en paralelo, la implementación es la misma.**



Cuando $SHIFT/\overline{LOAD}$ está en nivel BAJO (hay un 0), las compuertas AND de G_1 a G_3 se activan, y **permiten que cada bit del dato sea aplicado a la entrada D de sus respectivos flip-flops.**

Una vez se aplica un impulso de reloj, los FFD con D = 0 pasan al estado SET y los FFD con D = 1 pasan al estado RESET, almacenándose los 4 bits simultáneamente.

Por otro lado, cuando $SHIFT/\overline{LOAD}$ está en nivel ALTO (hay un 1), las compuertas AND de G_1 a G_4 se inhiben y las compuertas de G_5 a G_7 se activan, **permitiendo que los bits de datos se desplacen hacia la derecha**, pasando de un flip-flop al siguiente.

En cuanto a las **compuertas OR**, estas **permiten el desplazamiento normal** o la introducción de datos en paralelo, dependiendo de la salida que arrojen las compuertas AND que sirven de entrada a las OR, que, a su vez, dependen del estado de la entrada $SHIFT/\overline{LOAD}$.

El código de este módulo de control en Verilog es la siguiente.

```
1   /* PROYECTO:       FDD_ParallelInParallelOut
2
3      ARCHIVO:        FDD_Ctrl.v
4      ASIGNATURA:     DSD
5      DESCRIPCION:    Descripción de modulo que controla el comportamiento de un FFD,
6                      en términos de entrada en paralelo
7
8      PROF:           Flores Escobar Jose Antonio
9      EQUIPO:         Álvarez Hernández Gabriel Alexander
10                     Bueno Aguilar Alexis Haziel
11                     Huesca Laureano Josue Alejandro
12                     García Quiroz Gustavo Ivan
13                     Muñoz Valdivia Irving Omar
14                     Pedroza Villagomez Emir
15  */
16
17  // Defínción de puertos
18  module FDD_Ctrl(
19     input     serialin_i,
20     input     move_i,           // Entrada NOT para mover
21     input     move_ni,          // Entrada NOT para cargar (~move_i)
22     input     parallelin_i,
23     input     clk_i,
24     input     rst_i,
25     output    q_o
26  );
27     // Salida del 1er AND
28     wire      andoutft1_w;
29     assign    andoutft1_w = serialin_i & move_ni;
30
31     // Salida del 2do AND
32     wire      andoutft2_w;
33     assign    andoutft2_w = move_i & parallelin_i;
34
35     // Salida OR
36     wire      orout_w;
37     assign    orout_w = andoutft1_w | andoutft2_w;
38  endmodule
```

FFD_FPGA.v

Finalmente, este archivo es básicamente el registro de desplazamiento.

Este contiene instancias de FFDs interconectados con sus respectivas entradas y salidas, para formar dicho registro con entradas y salidas en paralelo.

Se incluye también un **banco de pruebas (testbench**) para la posterior simulación de las señales de este registro.

```
/* PROYECTO:       FDD_ParallelInParallelOut

   ARCHIVO:        FDD_FPGA.v
   ASIGNATURA:     DSD
   DESCRIPCION:    Descripción de modulo que implementa un registro de FFD
                   Parallel-In, Parallel-Out

   PROF:           Flores Escobar Jose Antonio
   EQUIPO:         Álvarez Hernández Gabriel Alexander
                   Bueno Aguilar Alexis Haziel
                   Huesca Laureano Josue Alejandro
                   García Quiroz Gustavo Ivan
                   Muñoz Valdivia Irving Omar
                   Pedroza Villagomez Emir
*/

// Defincion de puertos
module FFD_FPGA(
   input           clk_i,
   input           rst_i,
   input    [3:0] d_i,           // Arreglo de 4 entradas para cada FFD
   input           serialin_i,
   input           move_load_i,
   output wire     q0_o,          // Cada una de las señales de salida
   output wire     q1_o,
   output wire     q2_o,
   output wire     q3_o
);

// Declaración de señales
   wire     ff1q_ff2d,
            ff2q_ff3d,
            ff3q_ff4d,
            clk_df,
            salidactrl_w3,
            salidactrl_w2,
            salidactrl_w1,
            salidactrl_w0;


// Instancia del Divisor de Frecuencia
   DivFreq DF(
      .clk_i   (clk_i),
      .rst_i   (rst_i),
      .clk_o   (clk_df)
   );

// INSTANCIA FDD3
// Instancia del control del FFD 3
   FDD_Ctrl FDD_Ctrl3(
      .serialin_i    (serialin_i),
      .move_i        (move_load_i),
      .move_ni       (~move_load_i),
      .parallelin_i  (d_i[3]),
      .clk_i         (clk_i),
      .rst_i         (rst_i),
      .q_o           (salidactrl_w3)
   );

// Instancia del modulo para el FFD 3
   FFD FDD3(
//    .clk_i         (clk_df),            // Para FPGA
      .clk_i         (clk_i),             // Para simulación
      .d_i           (salidactrl_w3),
      .rst_i         (rst_i),
      .q_o           (ff1q_ff2d)
   );
```

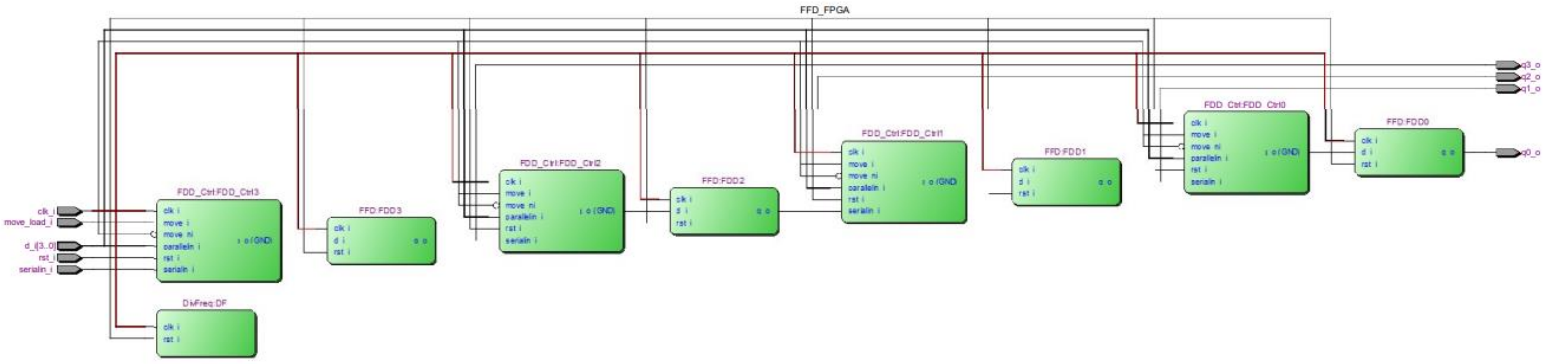
```
67      assign            q3_o = ff1q_ff2d;
68
69    // INSTANCIA FDD2
70    // Instancia del control del FFD 2
71      FDD_Ctrl FDD_Ctrl2(
72        .serialin_i     (ff1q_ff2d),
73        .move_i         (move_load_i),
74        .move_ni        (~move_load_i),
75        .parallelin_i   (d_i[2]),
76        .clk_i          (clk_i),
77        .rst_i          (rst_i),
78        .q_o            (salidactrl_w2)
79      );
80
81    // Instancia del modulo para el FFD 2
82      FFD FDD2(
83    //    .clk_i          (clk_df),             // Para FPGA
84        .clk_i          (clk_i),              // Para simulación
85        .d_i            (salidactrl_w2),
86        .rst_i          (rst_i),
87        .q_o            (ff2q_ff3d)
88      );
89      assign            q2_o = ff2q_ff3d;
90
91    // INSTANCIA FDD1
92    // Instancia del control del FFD 1
93      FDD_Ctrl FDD_Ctrl1(
94        .serialin_i     (ff2q_ff3d),
95        .move_i         (move_load_i),
96        .move_ni        (~move_load_i),
97        .parallelin_i   (d_i[1]),
98        .clk_i          (clk_i),
99        .rst_i          (rst_i),
100       .q_o            (salidactrl_w1)
101     );
102
103   // Instancia del modulo para el FFD 1
104     FFD FDD1(
105   //    .clk_i          (clk_df),             // Para FPGA
106       .clk_i          (clk_i),              // Para simulación
107       .d_i            (salidactrl_w1),
108       .rst_i          (rst_i),
109       .q_o            (ff3q_ff4d)
110     );
111     assign            q1_o = ff3q_ff4d;
112
113   // INSTANCIA FDD0
114   // Instancia del control del FFD 0
115     FDD_Ctrl FDD_Ctrl0(
116       .serialin_i     (ff3q_ff4d),
117       .move_i         (move_load_i),
118       .move_ni        (~move_load_i),
119       .parallelin_i   (d_i[0]),
120       .clk_i          (clk_i),
121       .rst_i          (rst_i),
122       .q_o            (salidactrl_w0)
123     );
124
125   // Instancia del modulo para el FFD 0
126     FFD FDD0(
127   //    .clk_i          (clk_df),             // Para FPGA
128       .clk_i          (clk_i),              // Para simulación
129       .d_i            (salidactrl_w0),
130       .rst_i          (rst_i),
131       .q_o            (q0_o)
132     );

133   endmodule
134
135   //*************************************************************************************
      *****//
```

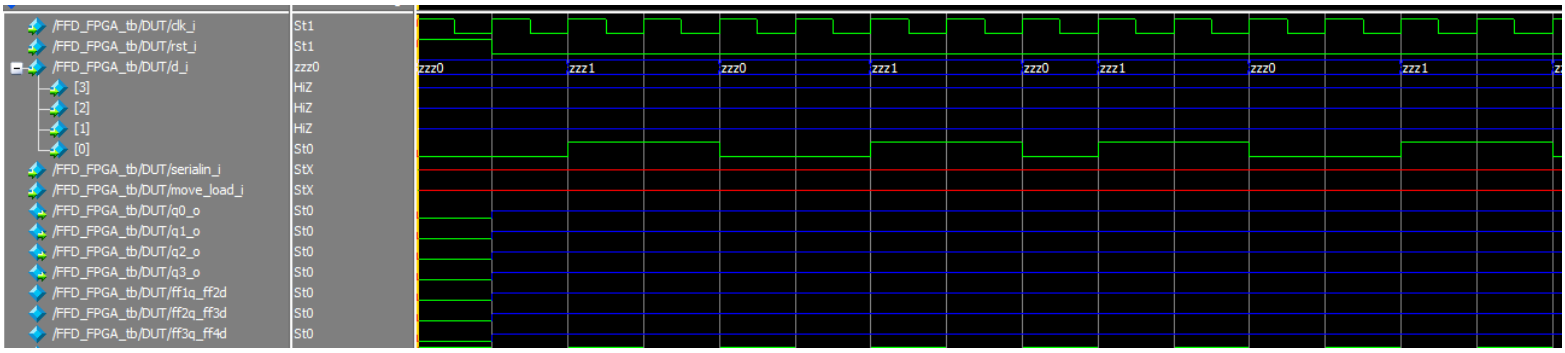
```
136
137  // Creando un banco de pruebas (testbench)
138  module FFD_FPGA_tb();
139     reg             clk_i;
140     reg             rst_i;
141     reg             d_i;
142     reg             serialin_i;
143     reg             move_load_i;
144     wire            q0_o;
145     wire            q1_o;
146     wire            q2_o;
147     wire            q3_o;
148
149     // Valores iniciales
150     initial
151     begin
152        clk_i     <= 1'b1;
153        rst_i     <= 1'b1;
154        d_i       <= 1'b0;
155        #100
156           rst_i <= 1'b0;
157     end
158
159     // Instancia del Device Under Test (DUT)
160     FFD_FPGA DUT(
161        .clk_i          (clk_i),
162        .d_i            (d_i),
163        .rst_i          (rst_i),
164        .serialin_i     (serialin_i),
165        .move_load_i    (move_load_i),
166        .q_0o           (q_0o),
167        .q_1o           (q_1o),
168        .q_2o           (q_2o),
169        .q_3o           (q_3o)
170     );
171
172     // Generacion de la señal de reloj
173     always
174     begin
175        #50
176           clk_i <= ~clk_i;
177     end
178
179     // Cambios de valores en señales
180     always
181     begin
182        #100
183           d_i    <= 1'b0;
184        #100
185           d_i    <= 1'b1;
186        #100
187           d_i    <= 1'b1;
188        #100
189           d_i    <= 1'b0;
190        #100
191           d_i    <= 1'b0;
192        #100
193           d_i    <= 1'b1;
194        #100
195           d_i    <= 1'b1;
196     end
197  endmodule
```

RTL VIEWER.

Una vez compilado todo el proyecto de manera exitosa, podemos ver el RTL resultante de nuestro registro.



SIMULACIÓN.



CONCLUSIONES

En esta práctica se comprendió una de muchas aplicaciones que pueden tener los flip-flops en los circuitos secuenciales, en este caso, hablando específicamente del flip-flop tipo D, utilizado en un registro de desplazamiento, ya que posee solo una entrada de datos y tiene la capacidad de almacenar datos binarios que entren en dicha línea.

Además, se analizaron las diferentes formas de implementar un registro en función de como se quieren recibir las entradas y las salidas: en serie o en paralelo. Cada una posee inclusive aplicaciones en la vida real. Para los registros con entrada y salida en paralelo, se vio el funcionamiento que tiene, incluyendo la implementación de un módulo de control para la entrada de datos en paralelo.

Al final, esta práctica sirvió de introducción a los circuitos secuenciales, al manejo en los flip-flops en aplicaciones prácticas en el entendimiento de la carga y desplazamiento de datos en un registro.

REFERENCIAS

Floyd. (2007). Fundamentos de Sistemas Digitales 9 Edición. Pearson Educacion.