Laboratorio 3

Macarena Oyague
Arquitectura de Computadores

May 26, 2020

Pregunta 2

Para la implementación de este ejercicio, se ha utilizado clrn en función a la imagen del esquemático, en lugar de clr que está establecido en la indicación.

El diseño está conformado por:

- m1: MUX 2:1
- m2: MUX 2:1
- m3: MUX 2:1
- d1: D Flip Flop
- d2: D Flip Flop
- d3: D Flip Flop

Fue desarrollado de manera Structural:

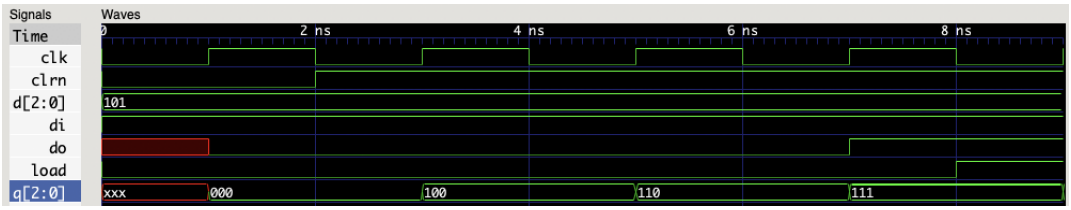
```
module SchematicMisterioso(   input [2:0]d,
                              input di,
                              input load,
                              input clk,
                              input clrn,
                              output [2:0]q,
                              output do);
    wire mout1, mout2, mout3;
    mux2_1 m1(.d_0(di), .d_1(d[2]), .selector(load), .out(mout1));
    DFF d1(.d(mout1), .clk(clk), .clrn(clrn), .q(q[2]));
    mux2_1 m2(.d_0(q[2]), .d_1(d[1]), .selector(load), .out(mout2));
    DFF d2(.d(mout2), .clk(clk), .clrn(clrn), .q(q[1]));
    mux2_1 m3(.d_0(q[1]), .d_1(d[0]), .selector(load), .out(mout3));
    DFF d3(.d(mout3), .clk(clk), .clrn(clrn), .q(q[0]));
    assign do = q[0];
endmodule
```

Donde los D Flip Flop son síncronos al clk.

Se utilizará un ejemplo con d y di como valores constantes con la finalidad de desarrollar una explicación correspondiente al funcionamiento. A partir de ello, se puede observar los inputs de entrada junto con los outputs de salida del esquemático misterioso:

```
 0 ns:  d=101  di=1  load=0  clk=0  clrn=0  q=xxx  do=x
 1 ns:  d=101  di=1  load=0  clk=1  clrn=0  q=000  do=0
 2 ns:  d=101  di=1  load=0  clk=0  clrn=1  q=000  do=0
 3 ns:  d=101  di=1  load=0  clk=1  clrn=1  q=100  do=0
 4 ns:  d=101  di=1  load=0  clk=0  clrn=1  q=100  do=0
 5 ns:  d=101  di=1  load=0  clk=1  clrn=1  q=110  do=0
 6 ns:  d=101  di=1  load=0  clk=0  clrn=1  q=110  do=0
 7 ns:  d=101  di=1  load=0  clk=1  clrn=1  q=111  do=1
 8 ns:  d=101  di=1  load=1  clk=0  clrn=1  q=111  do=1
 9 ns:  d=101  di=1  load=1  clk=1  clrn=1  q=101  do=1
```

El cual tiene como WaveForm lo siguiente:



Es posible observar que al ser 0 el clk, los outputs no varían. Además, el clrn sirve para settear los outputs en 0. Es posible notar que cuando load es 0, q y do tendrán que tomar el valor de di. Sin embargo, esto ocurre de una manera consecutiva puesto que la distribución interna behavioral depende de inputs previos. Por otro lado, cuando load es 1, no ocurre lo mismo y q y do tienen el valor que les corresponde sin depender de alguna iteración, únicamente del clk porque el D FlipFlop final es el que arrojará el output.

A continuación se podrá ver el comportamiento de los componentes internos, lo cual demostrará lo mencionado anteriormente. Las entradas serán llamadas inputs por motivos de simplicidad, sin embargo en el detalle de cada módulo se pueden ver las entradas que corresponden a cada una de ellas. El output q está asociado a los D Flip Flops como se ve en la imagen, y do está asociado a q[0].

Componente	Input 1	Input 2	Input 3	Output
m1	1	1	0	1
d1	x	0	0	x
m2	x	0	0	x
d2	x	0	0	x
m3	x	1	0	x
d3	x	0	0	x

- q = xxx
- do = x

La sensibilidad mayor en el circuito se da por clk, si clk está en 1 el output podrá cambiar porque, como se comentó, está relacionado al diseño del DFF. Como en este caso está en 0, no ocurre ninguna asignación por parte de d1, y los demás muxes no tienen que seleccionar en el 0 por estar x ahí.

Componente	Input 1	Input 2	Input 3	Output
m1	1	1	0	1
d1	1	1	0	0
m2	x	0	0	x
d2	x	1	0	0
m3	x	1	0	x
d3	x	1	0	0

- q = 000
- do = 0

Como clrn es 0, los outputs de los DFF son 0, ya que como el clk es 1 pueden ser asignados. Podemos ver que, como los inputs dependen de la asignación en la iteración anterior, el input que antes era x para d1 ya tiene la asignación del output de m1. Como el selector es 0, ya existe un output para arrojar.

Componente	Input 1	Input 2	Input 3	Output
m1	1	1	0	1
d1	1	0	1	0
m2	0	0	0	0
d2	x	0	1	0
m3	0	1	0	0
d3	x	0	1	0

- q = 000
- do = 0

Debido a que el primer input del m2 y m3 dependen de la asignación del output del DFF previo, ahora pueden tener una asignación en esta iteración. Pero como el clk es 0, no hay cambio en ningún output.

Componente	Input 1	Input 2	Input 3	Output
m1	1	1	0	1
d1	1	1	1	1
m2	0	0	0	0
d2	0	1	1	0
m3	0	1	0	0
d3	0	1	1	0

- q = 100
- do = 0

Como clk ascendió, el valor del output de d1 se volvió igual al valor d de entrada, que ahora puede settear porque clrn es 1.

Componente	Input 1	Input 2	Input 3	Output
m1	1	1	0	1
d1	1	0	1	1
m2	1	0	0	1
d2	0	0	1	0
m3	0	1	0	0
d3	0	0	1	0

- q = 100

- do = 0

Como el output de d1 varió, el input correspondiente al 0 en selector de m2 varió y con ello su output.

Componente	Input 1	Input 2	Input 3	Output
m1	1	1	0	1
d1	1	1	1	1
m2	1	0	0	1
d2	1	1	1	1
m3	0	1	0	0
d3	0	1	1	0

- q = 110
- do = 0

Ocurrirá lo mismo con d2.

Componente	Input 1	Input 2	Input 3	Output
m1	1	1	0	1
d1	1	0	1	1
m2	1	0	0	1
d2	1	0	1	1
m3	1	1	0	1
d3	0	0	1	0

- q = 110
- do = 0

Que ocasionará otro cambio en el primer input y, junto con ello, en el output de m3.

Componente	Input 1	Input 2	Input 3	Output
m1	1	1	0	1
d1	1	1	1	1
m2	1	0	0	1
d2	1	1	1	1
m3	1	1	0	1
d3	1	1	1	1

- q = 111
- do = 1

Para finalmente asignar todos los bits de q y el único bit de do con el mismo valor que di.

Componente	Input 1	Input 2	Input 3	Output
m1	1	1	1	1
d1	1	0	1	1
m2	1	0	1	0
d2	1	0	1	1
m3	1	1	1	1
d3	1	0	1	1

- q = 111
- do = 1

Luego, al cambiar el selector a 1, habrá un cambio en el output de los muxes, pero q no variará porque clk está en 0.

Componente	Input 1	Input 2	Input 3	Output
m1	1	1	1	1
d1	1	1	1	1
m2	1	0	1	0
d2	0	1	1	0
m3	1	1	1	1
d3	1	1	1	1

- q = 101
- do = 1

Cuando hay un cambio ascendente en el clock, clrn está en 0 y load en 1, el output de los DFF será correspondiente a la entrada d del esquemático, lo generará que ésta se copie en q y el bit0 de ella se dirija a do.

Pregunta 3

El diseño del FIFO se realizo de manera Behavioral debido a los múltiples componentes que tiene. Estos son los siguientes:

- F0: D Flip Flop
- RS1: RS Latch
- RS2: RS Latch
- RS3: RS Latch
- RS4: RS Latch
- R1: Register
- R2: Register
- R3: Register
- R4: Register

Incluyendo además el setteo de los clks, se puede observar la implementación de la siguiente manera:

```
module FIFO4(   input [3:0]din,
                input read,
                input write,
                input clr,
                output [3:0]dout,
                output empty,
                output full);
    wire clk1, clk2, clk3, clk4;
    wire a4, an4, a3, an3, a2, an2, a1, an1, a0;
    assign clk4 = ~read & an4 & a3;
    assign clk3 = an3 & a2;
    assign clk2 = an2 & a1;
    assign clk1 = an1 & a0;
    DFF F0(~clr, write, clr | clk1, a0);
    RSLatch RS1(clk1, clk2, clr, a1, an1);
    RSLatch RS2(clk2, clk3, clr, a2, an2);
    RSLatch RS3(clk3, clk4, clr, a3, an3);
    RSLatch RS4(clk4, read, clr, a4, an4);
    wire [3:0]d12, d23, d34;
    Register R1(din, clk1, d12);
    Register R2(d12, clk2, d23);
    Register R3(d23, clk3, d34);
    Register R4(d34, clk4, dout);
endmodule
```

El FIFO funciona de la manera en la que cada Register podrá servir para almacenar data. Al estar write encendido se escribirá la data en los registros. Esta data será "empujada" hasta el cuarto registro y luego hacia el output, si se realiza write de otra data, esta irá al registro tres, si se realiza nuevamente

se dirigirá al registro 2 y si se realiza write nuevamente al registro 1. Se realiza de una manera encadenada, por lo que al ser empujado el input, si es que no se ha acumulado aún, la data permanecerá igual que el siguiente registro al que se empujo.

El output empty será 1 es que el cuarto registro está vacío (y ya ha sido leído), y el output full será 1 si es que el primer registro está lleno con con una data distinta a los demás registros (que no haya sido la empujada/leída). Dout siempre será asignado en función al R4 y al clk4 que le corresponde. Como se mencionó, mientras se realice otro write, se irá acumulando en cada registro a manera de stack, por lo cual es que habrá un punto en el que estará lleno (full).

Al prenderse read, por otro lado, lo que ocurrirá es que se vaciará el cuarto registro y empujará la data. Cada read hará una iteración de esta manera, lo cuál puede llegar a ocasionar que empty esté en 1 según la explicación dada.