Guía Práctica 1

Ejercicio 1 - Sugerido.

- Escribir el código Verilog para implementar el diseño de la Figura 1 considerando reset asíncrono.
- Genere la señal de reset apropiada para el registro de realimentación utilizado en el diseño.
- Escribir un testbench que permita verificar el correcto funcionamiento del circuito propuesto. Los estímulos pueden ser generados con python o modelados en el testbench.
- Cuantos ciclos de reloj son necesarios para que el registro o_data produzca overflow cuando i_sel, i_data1 e i_data2 son iguales a 1?.

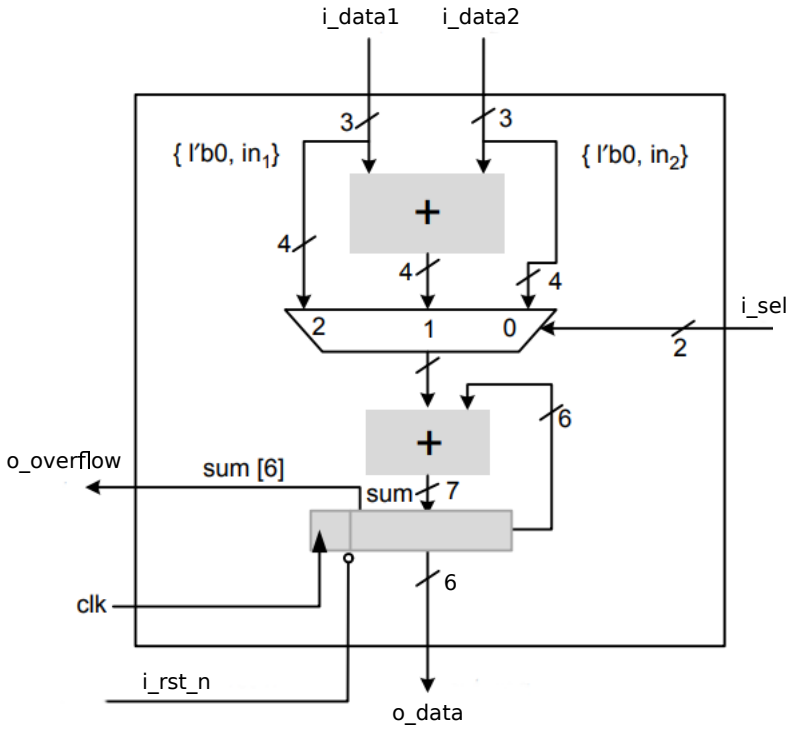

Figura 1: Diseño digital a nivel RTL con registro de realimentación

Ejercicio 2.

- Realizar el esquemático o diagrama en bloque del datapath de un selector de operaciones que ejecuta las siguientes operaciones aritméticas en paralelo con dos entradas i_dataA e i_dataB de tipo signadas de 16 bits y asigne el valor del resultado a una salida o_dataC de 16 bits.
- La elección de la operación a realizar depende de una señal de control i_sel de 2 bits.
- Implementar el diseño en Verilog y el testbench para verificar el comportamiento.

Operaciones:

$$\begin{aligned} o\_dataC &= i\_dataA + i\_dataB \\ o\_dataC &= i\_dataA - i\_dataB \\ o\_dataC &= i\_dataA \& i\_dataB \\ o\_dataC &= i\_dataA | i\_dataB \end{aligned} \tag{1}$$

Ejercicio 3.

Dibuje el esquemático para el siguiente código Verilog. Especifique de forma clara los tamaños de datos para todos los cables y muestre multiplexores, registros y señales de clock y reset.

```
module test_module(input [31:0]       x0,
                   input [1:0]        sel,
                   input              clk,
                   input              rst_n,
                   output reg [31:0] y0);

   reg [31:0]                         x1, x2, x3;
   reg [31:0]                         y1;
   wire [31:0]                        out;

   assign out = (x0 + x1 + x2 + x3 + y1)>>>2;
   always @(posedge clk or negedge rst_n)
     begin
        if(!rst_n) begin
           x1 <= 0;
           x2 <= 0;
           x3 <= 0;
        end
        else if (sel==0) begin
           x3 <= x2;
           x2 <= x1;
           x1 <= x0;
        end
        else if (sel == 01) begin
           x3 <= x1;
           x2 <= x0;
           x1 <= x2;
        end
        else begin
           x3 <= x3;
           x2 <= x2;
           x1 <= x0;
        end
     end

   always @ (posedge clk or negedge rst_n) begin
      if(!rst_n) begin
         y1 <= 0;
         y0 <= 0;
      end
      else begin
         y1 <= y0;
         y0 <= out;
      end
   end
endmodule
```

Ejercicio 4 - Sugerido.

- Dibujar la arquitectura de la siguiente ecuación diferencial
- Escribir el código Verilog
- Escribir el testbench para verificar el comportamiento

$$y[n] = x[n] - x[n-1] + x[n-2] + x[n-3] + 0{,}5y[n-1] + 0,25y[n-2] \tag{2}$$

Implemente las multiplicaciones de 0,5 y 0,25 mediante operaciones de desplazamiento.