

# 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?.

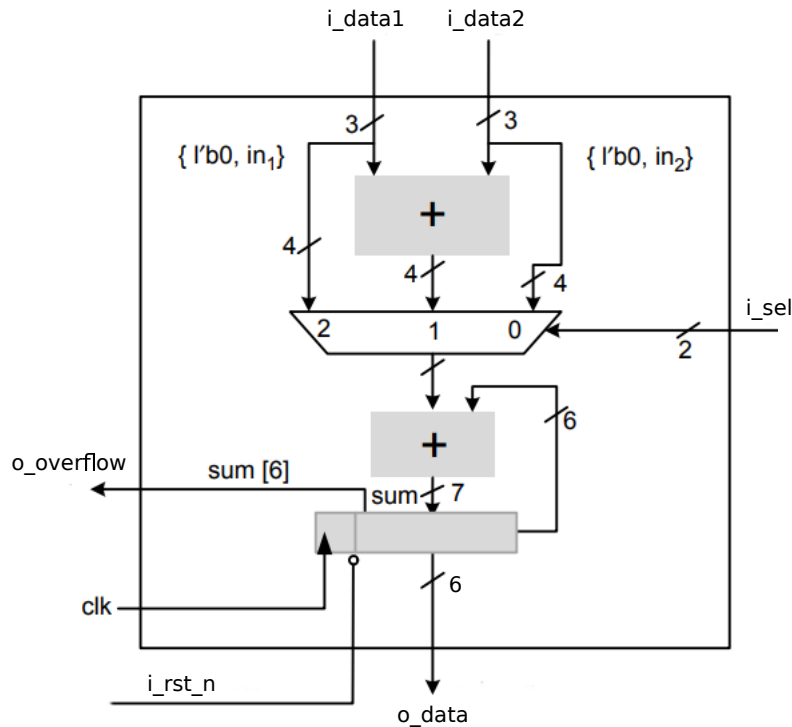


Figura 1: Diseño digital a nivel RTL con registro de realimentación

## 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:

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

## 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.

```

1 module test_module(input [31:0] x0,
2                     input [1:0] sel,
3                     input clk,
4                     input rst_n,
5                     output reg [31:0] y0);
6
7     reg [31:0] x1, x2, x3;
8     reg [31:0] y1;
9     wire [31:0] out;
10
11     assign out = (x0 + x1 + x2 + x3 + y1)>>>2;
12     always @(posedge clk or negedge rst_n)
13     begin
14         if (!rst_n) begin
15             x1 <= 0;
16             x2 <= 0;
17             x3 <= 0;
18         end
19         else if (sel==0) begin
20             x3 <= x2;
21             x2 <= x1;
22             x1 <= x0;
23         end
24         else if (sel == 01) begin
25             x3 <= x1;
26             x2 <= x0;
27             x1 <= x2;
28         end
29         else begin
30             x3 <= x3;
31             x2 <= x2;
32             x1 <= x0;
33         end
34     end
35
36     always @ (posedge clk or negedge rst_n) begin
37         if (!rst_n) begin
38             y1 <= 0;
39             y0 <= 0;
40         end
41         else begin
42             y1 <= y0;
43             y0 <= out;
44         end
45     end
46 endmodule

```

**Ejercicio 4 - Sugerido.**

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

$$y[n] = x[n] - x[n - 1] + x[n - 2] + x[n - 3] + 0,5y[n - 1] + 0,25y[n - 2] \quad (2)$$

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