Laboratorio 1. Propuestas de diseño.

Estudiantes: Ana Melissa Vásquez Rojas
Darío Garro Maya.

Problema 1. Decodificador Bin – Gray.

Entrada: Número de 4 bits
Salida: código Gray

Propuesta 1:

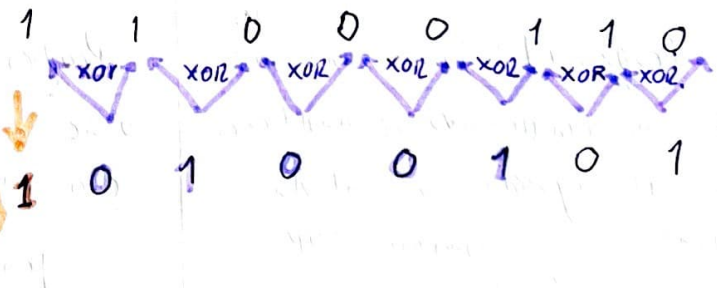
→ Mantener el bit más significativo igual
→ Cada bit siguiente se obtiene como XOR del bit actual con el bit más significativo anterior

XOR

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Ejemplo con un número de 8 bits.

$(11000110)_2 = (10100101)_{Gray}$



Propuesta 2: Hacer la conversión usando desplazamiento. y XOR.

→ Se realiza un desplazamiento a la derecha 1 posición (>> 1)
Ejemplo con un número de 8 bits
bin = 11000110

bin >> 1 = 01100011

→ Luego se hace un XOR entre el numero original y el desplazado.

Continua en la siguiente página...

Bit original	Bit desplazado	XOR.
1	0	1
1	1	0
0	1	1
0	0	0
0	0	0
1	0	1
1	1	0
0	1	1

↙ Este resultado corresponde al código gray

$(11000110)_2 = (10100101)_{gray}$ //

Comparación

Propuesta 1: Sumas sucesivas	Propuesta 2: Desplazamiento y XOR
→ Bit más significativo se mantiene. → Cada bit siguiente se calcula individualmente. usando XOR con el más significativo anterior → Es más intuitivo	→ Se calcula todo el código gray con la expresión. $gray = bin \oplus (bin >> 1)$ donde bin es el número binario de N bits. → Más compacta.

Ambas propuestas tienen una complejidad similar, elegimos la propuesta 1 porque es más intuitiva.

Problema 2:

Restador completo de 1bit

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Se busca la solución para S

A \ B Cin

	00	01	11	10
0		1		1
1	1		1	

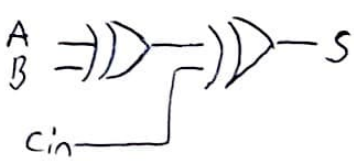
$A\bar{B}\bar{C_{in}} + \bar{A}\bar{B}C_{in} + ABC_{in} + \bar{A}B\overline{C_{in}}$

$S = C_{in}(\bar{A}\bar{B} + AB) + \overline{C_{in}}(A\bar{B} + \bar{A}B)$

$S = C_{in}(\overline{A \oplus B}) + \overline{C_{in}}(A \oplus B)$

$S = C_{in} \oplus (A \oplus B)$

↳



Se busca la solución para Cout:

A \ B Cin

	00	01	11	10
0		1	1	1
1			1	

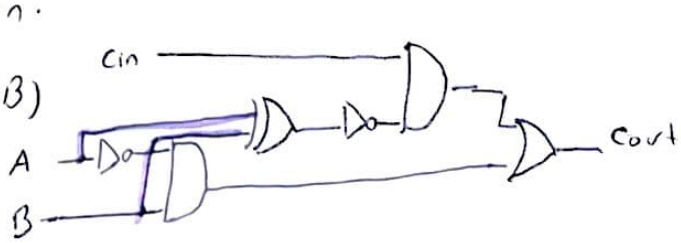
$C_{out} = BC_{in} + \bar{A}C_{in} + \bar{A}B$

$= BC_{in} + \bar{A}(C_{in} + B)$

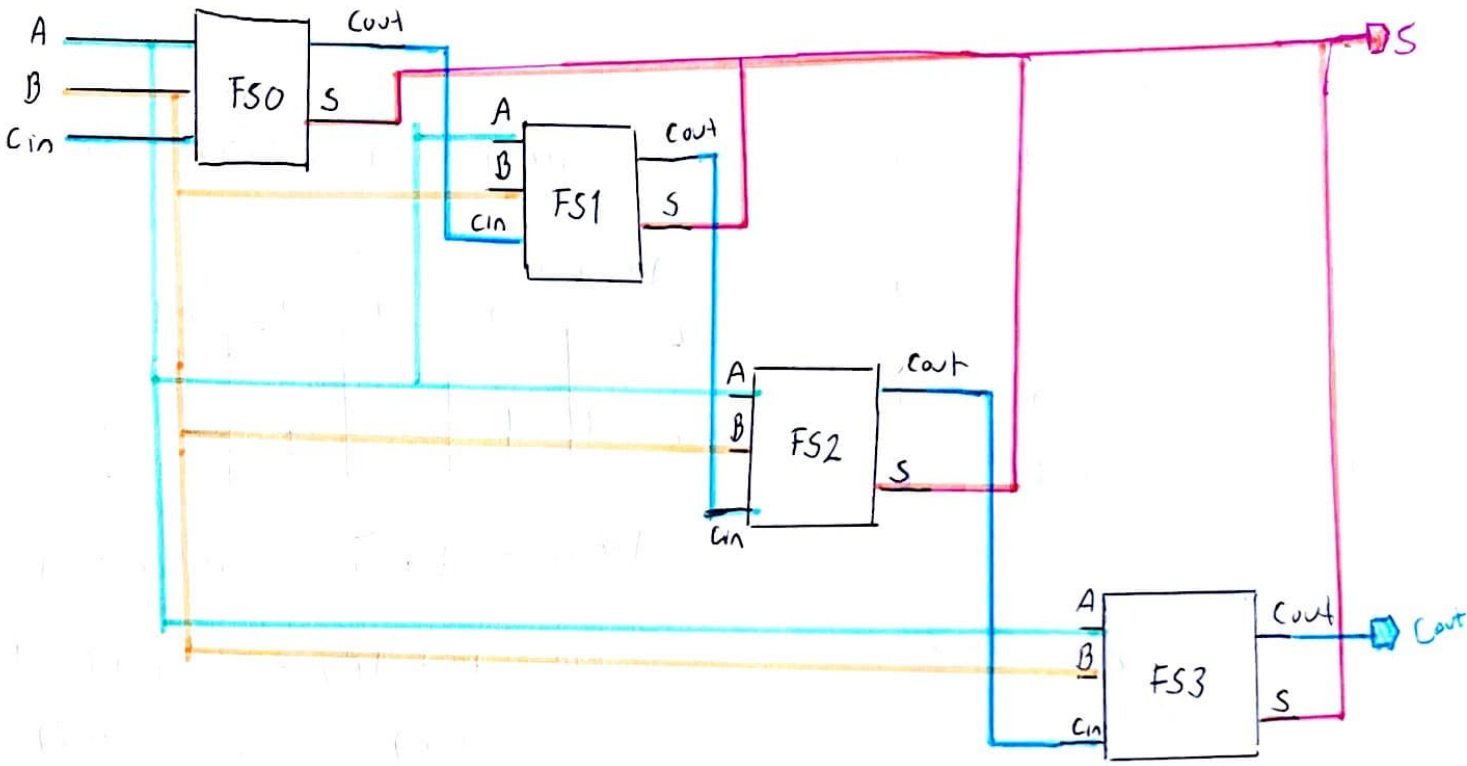
$C_{out} = \bar{A}\bar{B}C_{in} + \bar{A}B\bar{C_{in}} + \bar{A}BC_{in} + ABC_{in}$

$= \bar{A}B(C_{in} + \overline{C_{in}}) + C_{in}(\bar{A}\bar{B} + AB)$ ($C_{in} + \overline{C_{in}} = 1$)

$= \bar{A}B + C_{in}(\overline{A \oplus B})$



Restador completo 4 bits apartir de 1bit



Propuesta 1. Esta propuesta corresponde al diseño anterior utilizando las ecuaciones simplificadas para Cout y S para el restador completo de 1 bit.

$Cout = \bar{A}B + Cin\,(\overline{A \oplus B})$

$S = Cin \oplus (A \oplus B)$

Luego, a partir de ese hacer el de 4 bits

Propuesta 2. Esta propuesta corresponde a utilizar la ecuaciones de S y Cout sin simplificar (utilizando Cout extraído del mapa K.) para el restador completo de 1bit.

$Cout = \cdot\, BCin + \bar{A}Cin + \bar{A}B$

$S = A\bar{B}\bar{Cin} + \bar{A}\bar{B}Cin + ABCin + \bar{A}B\bar{Cin}$

y luego partir de ese para hacer el de 4bits

Al comparar ambas propuestas notamos que la propuesta 2 requiere utilizar muchas más compuertas en comparación a la propuesta 1.

El hecho de que en la propuesta 1 las ecuaciones estén simplificadas hace que se reduzca la cantidad de código necesario para resolver el problema en VHDL. Es por ello que la propuesta seleccionada es la 1.

Problema 3.

Propuesta 1) System Verilog

```
1   // Contador parametrizable de N bits con reset asíncrono
2   module param_counter #(
3       parameter int N = 6,                          // ancho (bits)
4       parameter logic [N-1:0] INIT = '0             // valor tras reset
5   ) (
6       input  logic             clk,                 // reloj
7       input  logic             arst,                // reset asíncrono, activo en 1
8       input  logic             en,                  // enable (1 = incrementa)
9       output logic [N-1:0]     q                    // salida
10  );
11
12      always_ff @(posedge clk or posedge arst) begin
13          if (arst) begin
14              q <= INIT;                            // reset inmediato (asíncrono)
15          end else if (en) begin
16              q <= q + 1'b1;                        // suma módulo 2^N
17          end
18          // si en=0, mantiene q
19      end
20
21  endmodule
```

Características

- Parametrizable: N fija el rango (p. ej., N=2 → 0..3; N=6 → 0..63).
- Valor inicial configurable: INIT.
- Sencillez y legibilidad: una sola ecuación: si en entonces q <= q+1.
- clk: Señal de reloj.
- arst: Reset asíncrono activo en alto (pone el contador a 0 sin esperar al flanco del reloj).
- en: Enable (si está en ‘1’, el contador incrementa; si está en ‘0’, mantiene el valor).
- q: Salida del contador.

Propuesta 2) VHDL

```vhdl
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity counter_n is
  generic (
    N    : positive := 6;                          -- ancho
    INIT : unsigned(N-1 downto 0) := (others => '0')
  );
  port (
    clk  : in  std_logic;
    arst : in  std_logic;                          -- reset asíncrono activo en '1'
    en   : in  std_logic;
    q    : out unsigned(N-1 downto 0)
  );
end entity;

architecture rtl of counter_n is
  signal q_r : unsigned(N-1 downto 0) := INIT;
begin
  q <= q_r;

  process (clk, arst)
  begin
    if arst = '1' then                             -- reset asíncrono
      q_r <= INIT;
    elsif rising_edge(clk) then
      if en = '1' then
        q_r <= q_r + 1;                            -- módulo 2^N por ancho
      end if;
    end if;
  end process;
end architecture;
```

Características

- Parametrización por generic (N, INIT).
- Uso de numeric_std para suma sobre unsigned.
- Reset asíncrono a nivel RTL.

Tabla de transición (común en ambas propuestas)

arst	en	Q (actual)	Q(siguiente)
1	X	XX	00
0	0	00	00
0	1	00	01
0	1	01	10
0	1	10	11
0	1	11	00

Ambas propuestas son correctas, equivalentes en funcionalidad y sintetizables. Sin embargo, SystemVerilog lo hace de manera más concisa y directa, por ejemplo, los tipos logic y bloques always_ff facilitan la lectura, siendo esta la propuesta elegida por su facilidad ante la otra propuesta a pesar de hacer lo mismo.