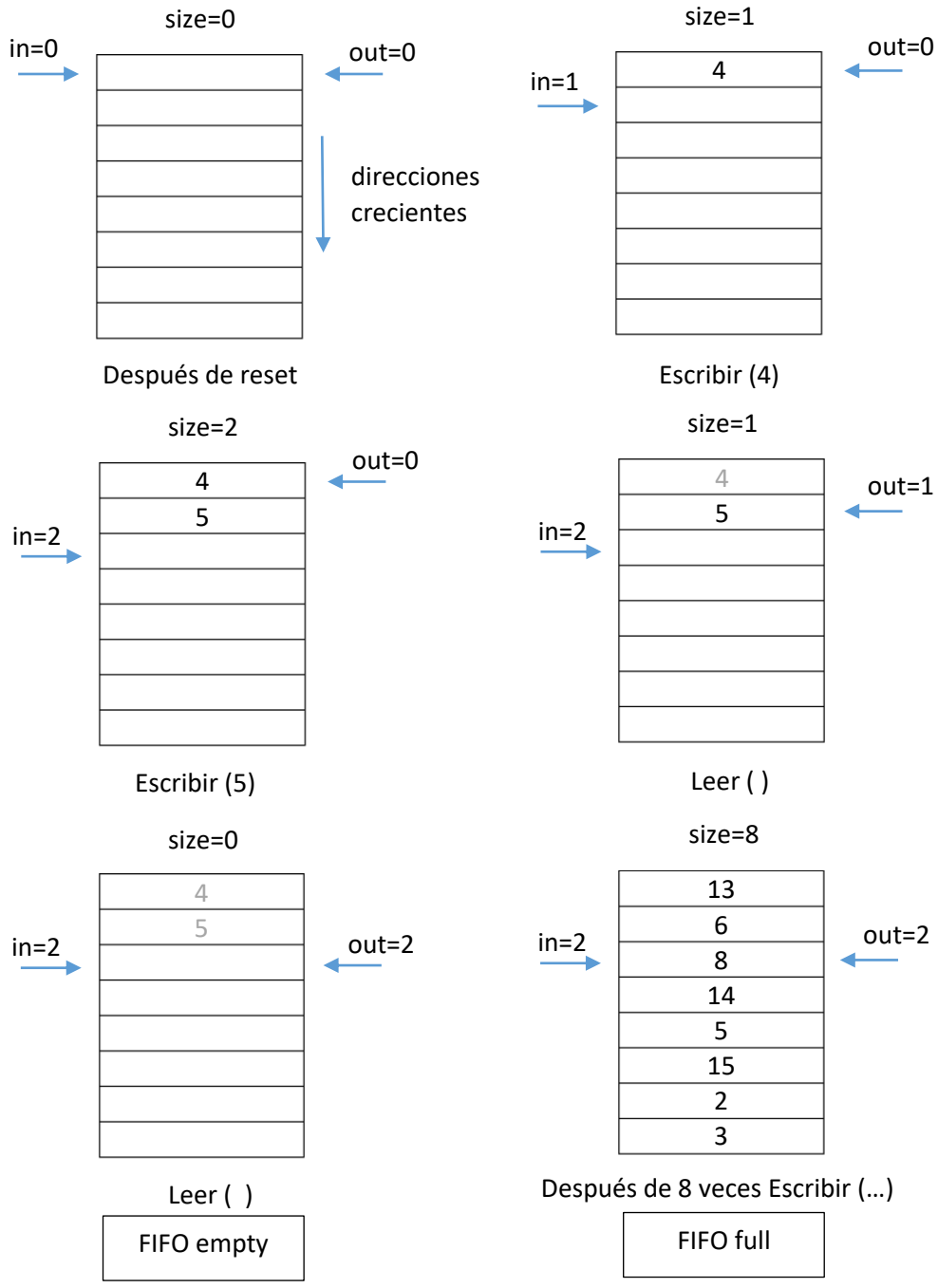
Nombre: EXAMEN Aprobación Directa 19/11/24

1) Describir en Verilog una memoria síncrona de tipo FIFO (First-Input-First-Output) con las siguientes especificaciones: 8 etapas o direcciones, 4 bits de tamaño de datos para **data_in** y **data_out**. Debe contar con una entrada síncrona de reset (**rst**) y las señales de "status" siguientes:
**full**: un uno cuando la FIFO está llena, de otro modo en cero.
**empty**: uno cuando la FIFO está vacía, de otro modo en cero.
**fifo_cnt**: contador de datos cargados en la FIFO (size).
El funcionamiento de la FIFO es de acuerdo al siguiente esquema que deberá implementarse:




2) A fin de asegurar la integridad de la cantidad de datos en la memoria, determine e implemente el código de Hamming para el registro size.
3) Cuando la memoria esté llena, el sistema deberá indicarlo haciendo parpadear un led al ritmo de una señal de clk de 1hz que se dispone. Diseñe el sistema secuencial correspondiente con una máquina de Moore.
4) En el punto anterior, No use condiciones sin cuidado, y en caso de tenerlas dele valor 0 e implemente la lógica combinacional de la salida que comanda el led con multiplexores de dos canales únicamente. Describa en Verilog esta lógica usando el operardor ternario.

RESPUESTA parte 1)

```
module fifo  (input      [3:0] data_in, //La memoria tiene 4 bits de dato
          input    clk, rst, rd, wr,
          output       empty, full,
          output reg [3:0] fifo_cnt, //El contador de datos ocupados por la FIFO
          output reg [3:0] data_out);

  reg [3:0] fifo_ram[7:0];  // Memoria de 8 x 4
  reg [2:0] rd_ptr, wr_ptr; // Punteros o índices para lectura y escritura. (in y out)

  assign empty = (fifo_cnt == 4'h0);
  assign full = (fifo_cnt == 4'h8);

  always @( posedge clk )
    if( rst )
     begin
      wr_ptr <= 0;    // in=0
      rd_ptr <= 0;    // out=0
      fifo_cnt <= 0;  // size=0
      data_out <= 0;
     end
    else
     begin
      if((wr && !full)||(wr && rd && !empty))  //Operación de escritura
        begin
          fifo_ram[wr_ptr] <= data_in;  //almaceno dato en la RAM
          wr_ptr <= wr_ptr + 1'b1;         //incremento in
          if (!(rd && !empty)) fifo_cnt <= fifo_cnt + 1'b1;  // incremento size
        end
      if(rd && !empty)  //Operación de lectura
        begin
          data_out <= fifo_ram[rd_ptr];
          rd_ptr <= rd_ptr + 1'b1;
          if (!wr) fifo_cnt <= fifo_cnt - 1'b1;
        end
     end
endmodule
```