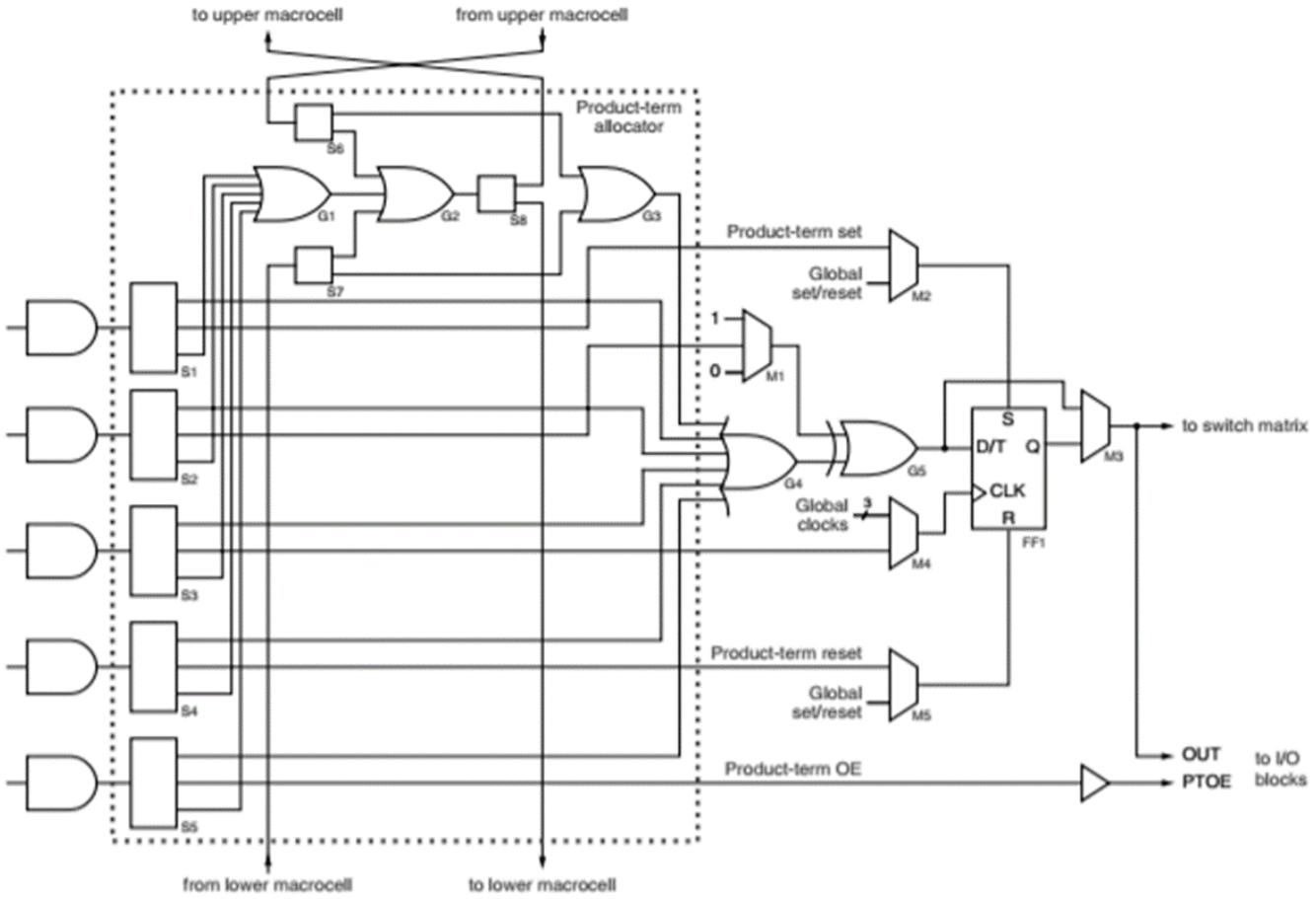
Electrónica III

Curso 2021

Preguntas sobre CPLD y FPGA

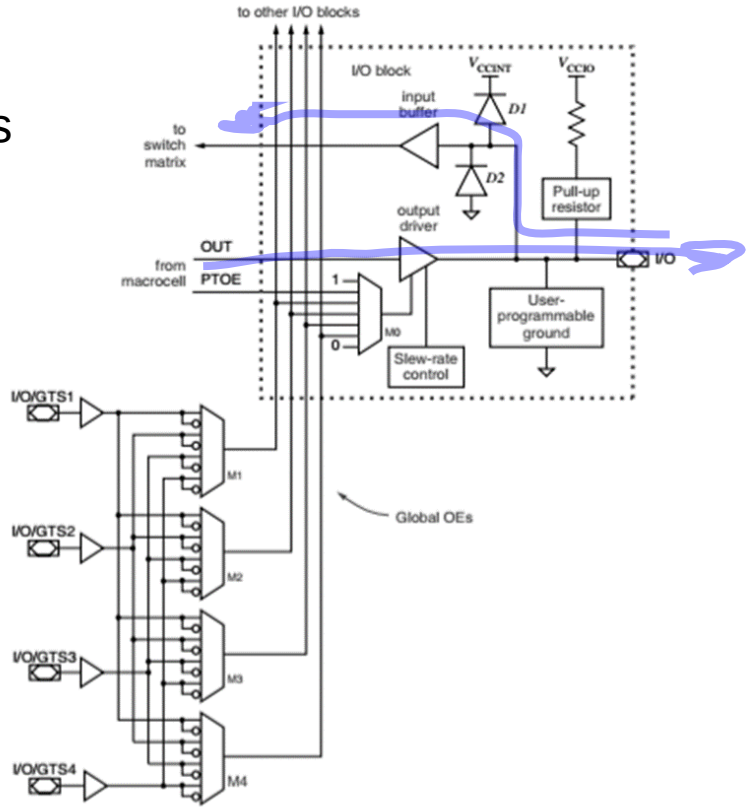
ITBA

Explique de manera clara, completa y concisa como opera un functional block de una CPLD



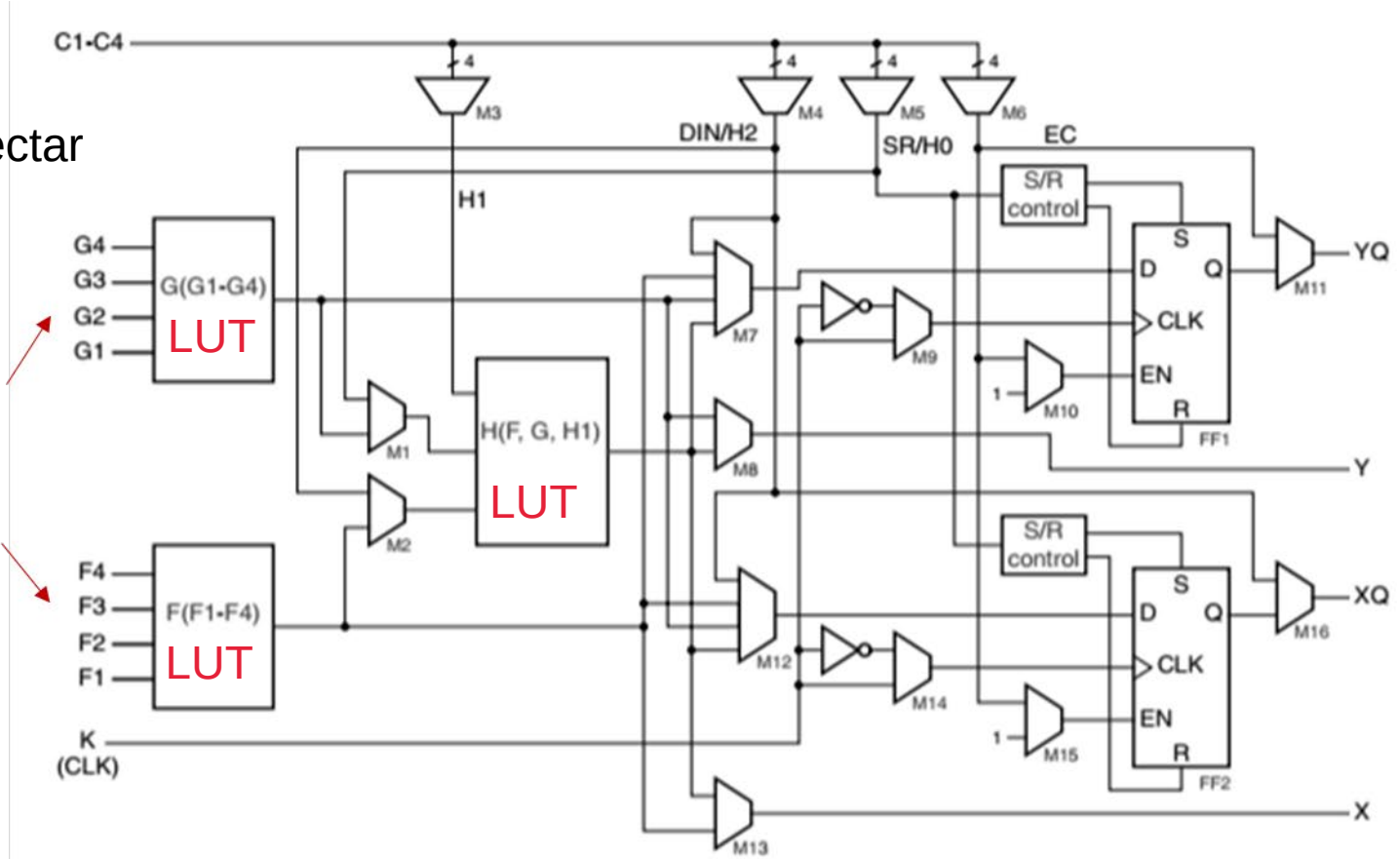
Explique de manera clara, completa y concisa como opera un I/O de una CPLD

* Entrada y salida three state
* Diodos de protección
* Pull-up o pull-down para salidas y entradas
* SlewRate control



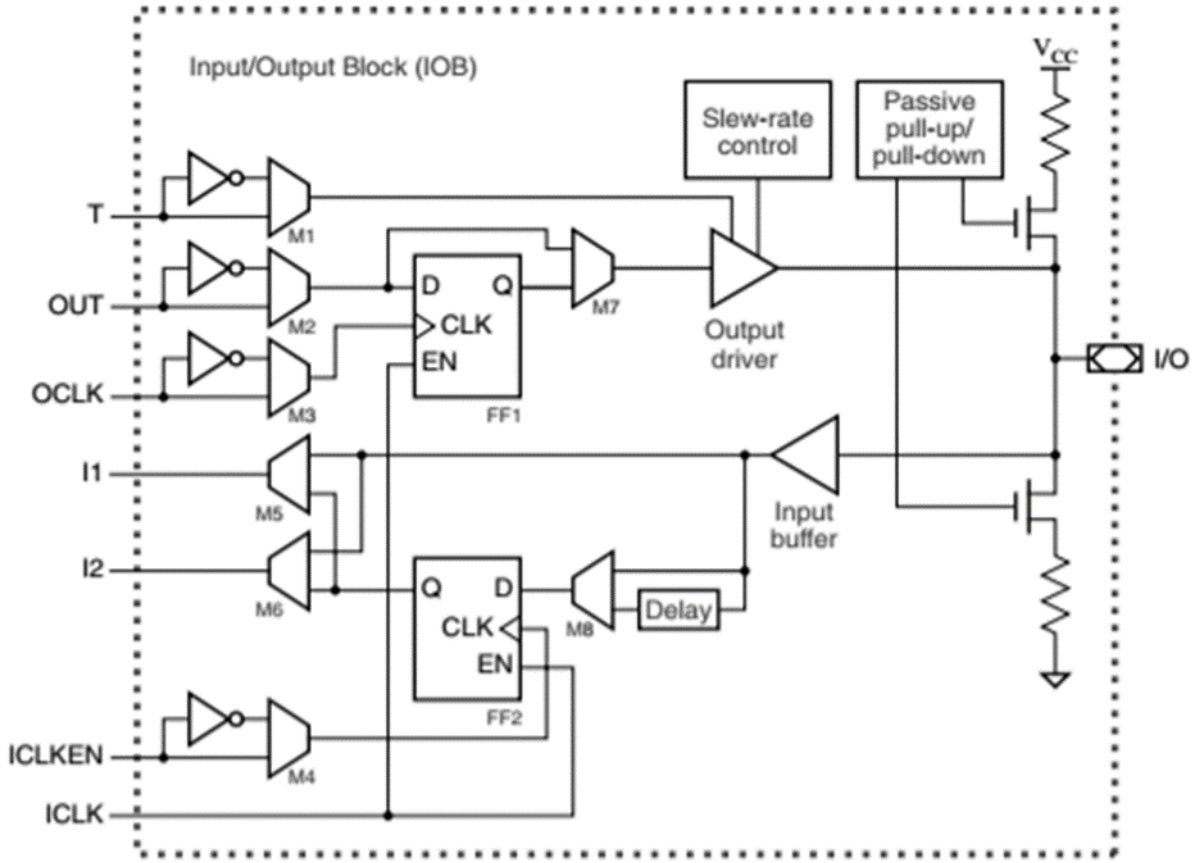
Explique de manera clara, completa y concisa como opera un programmable logic block de una FPGA

* Alta customizacion
* Se puede interconectar a mi gusto (como las empanadas)



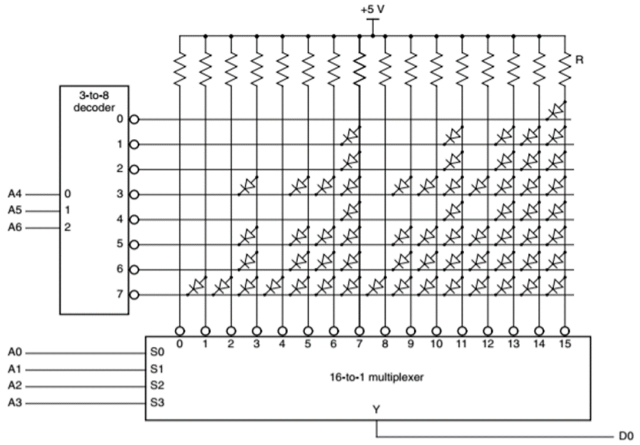
Pueden generar cualquier función Lógica de sus 4 Entradas

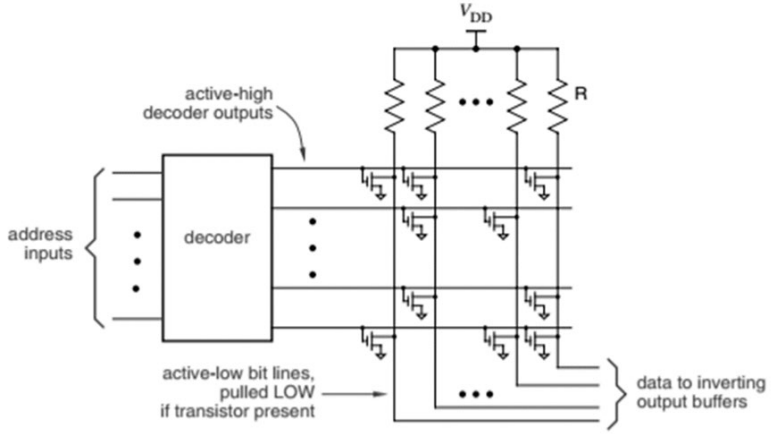
Explique de manera clara, completa y concisa como opera un I/O block de una FPGA



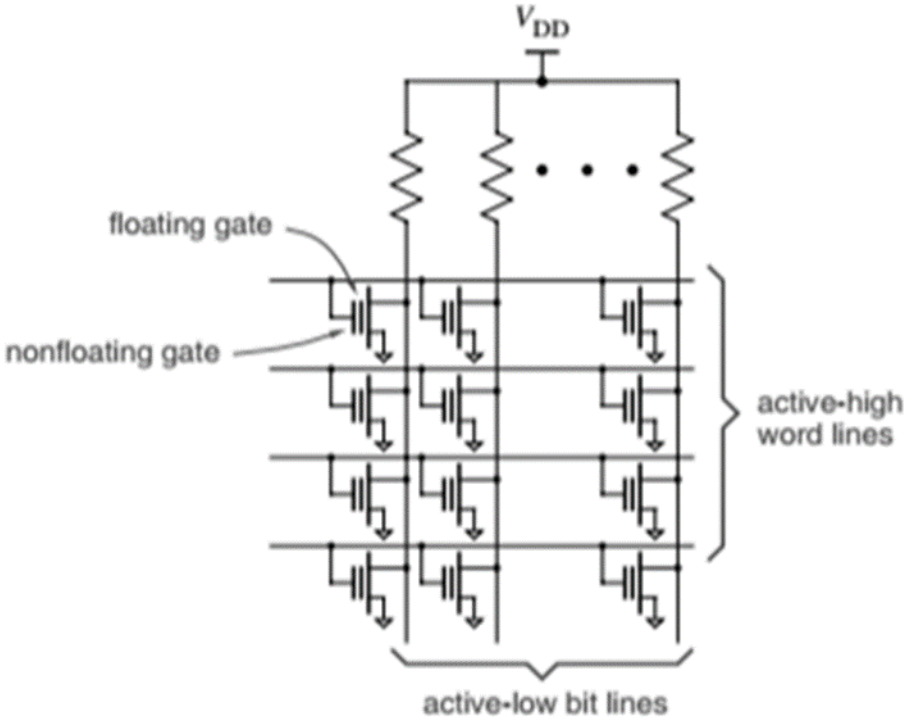
Preguntas sobre Memorias

Explique de manera clara, completa y concisa qué es y como funciona una PROM. Amplíe y agregue todos los diagramas que considere necesarios.

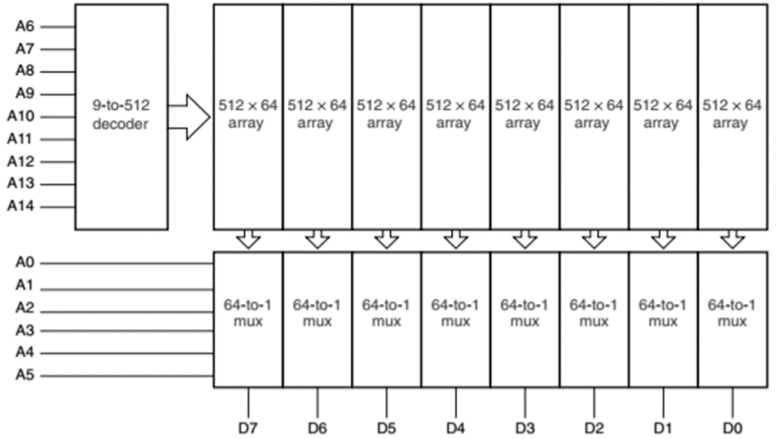


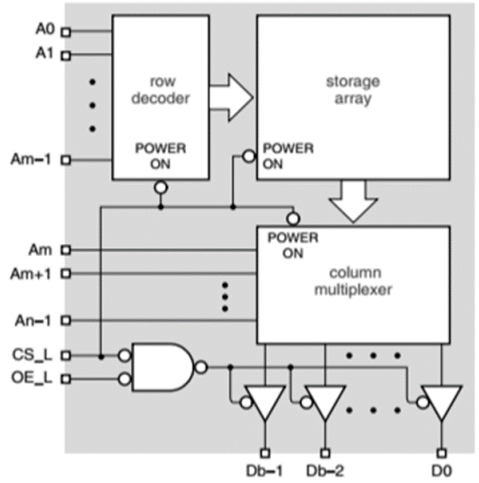


Explique de manera clara, completa y concisa qué es y como funciona una EPROM. Amplíe y agregue todos los diagramas que considere necesarios.

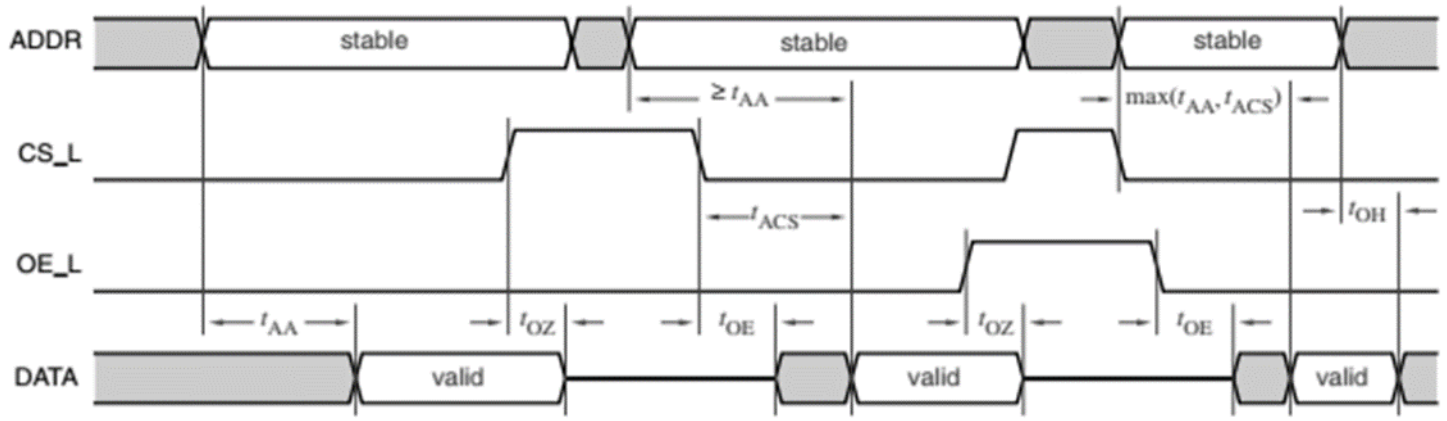


Explique de manera clara, completa y concisa qué es y como esta organizada una memoria de solo lectura. Amplíe y agregue todos los diagramas que considere necesarios.

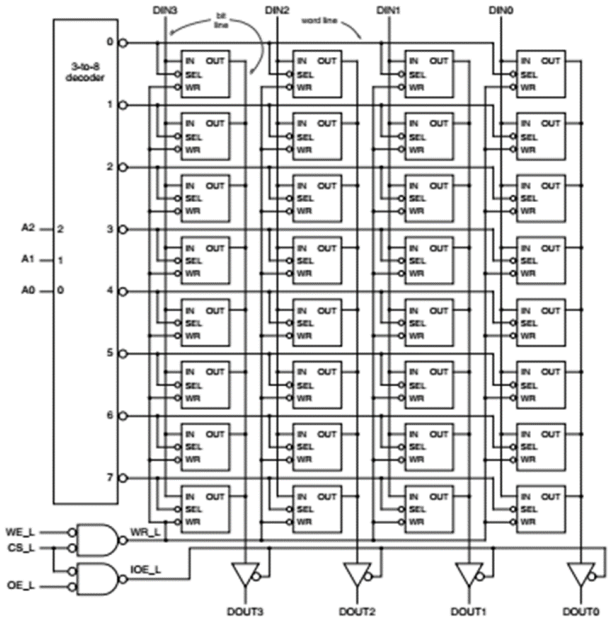


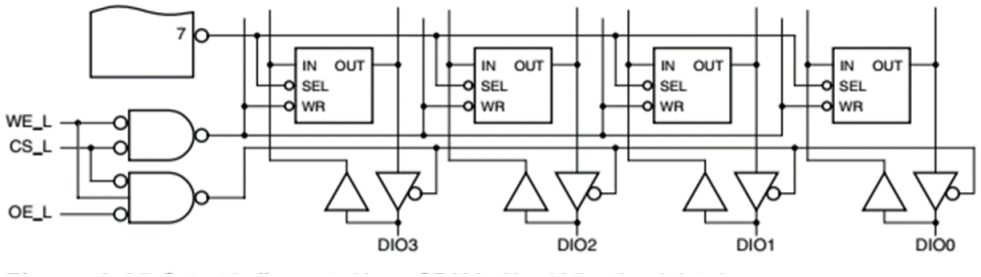


Explique de manera clara, completa y concisa qué es y como funciona el siguiente diagrama de tiempos en una memoria de solo lectura. Detalle los timming parameters. Amplíe y agregue todos los diagramas que considere necesarios.

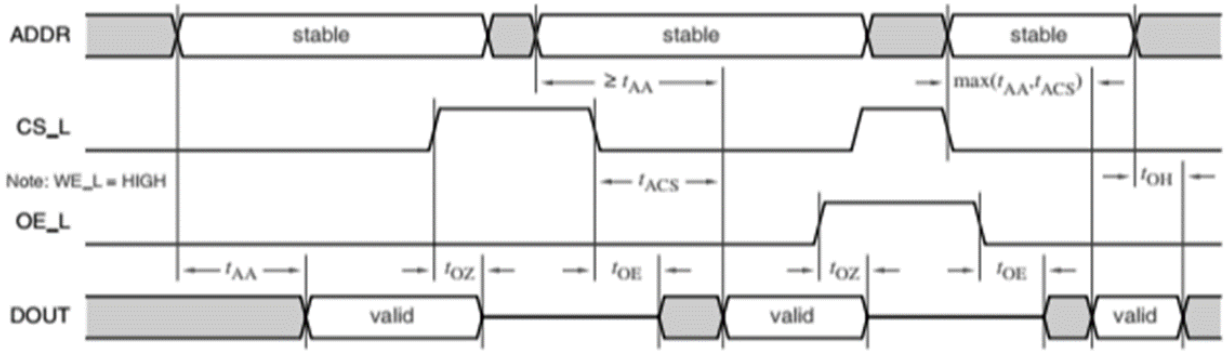


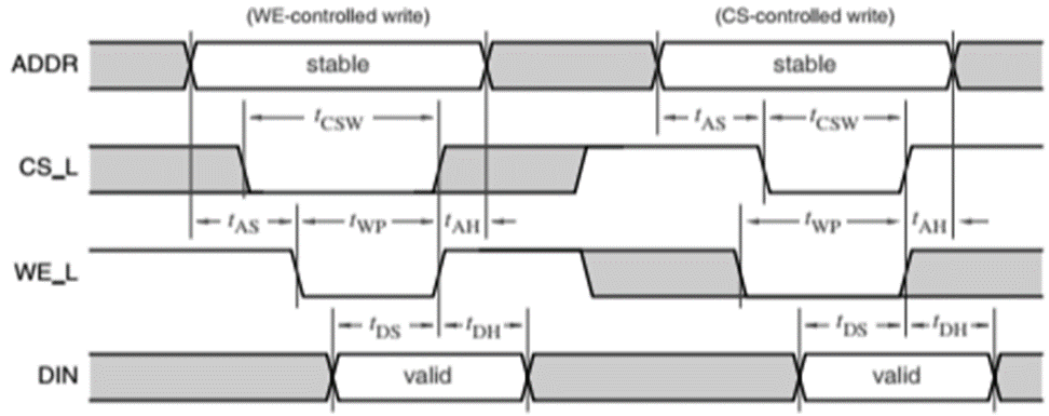
Explique de manera clara, completa y concisa qué es y como funciona una RAM Estática y como operan los siguientes diagramas en bloques. Amplíe y agregue todos los diagramas que considere necesarios.



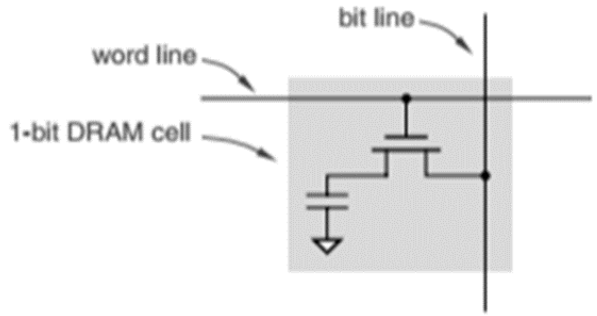


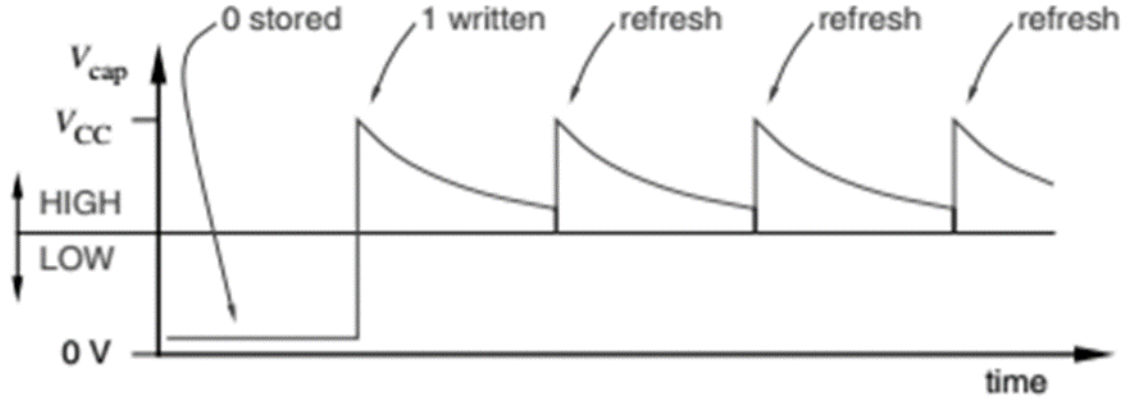
Explique de manera clara, completa y concisa qué es y como funciona el siguiente diagrama de tiempos en una memoria de solo lectura. Detalle los timming parameters. Amplíe y agregue todos los diagramas que considere necesarios.





Explique de manera clara, completa y concisa qué es y como funciona una RAM Dinámica. Amplíe y agregue todos los diagramas que considere necesarios.

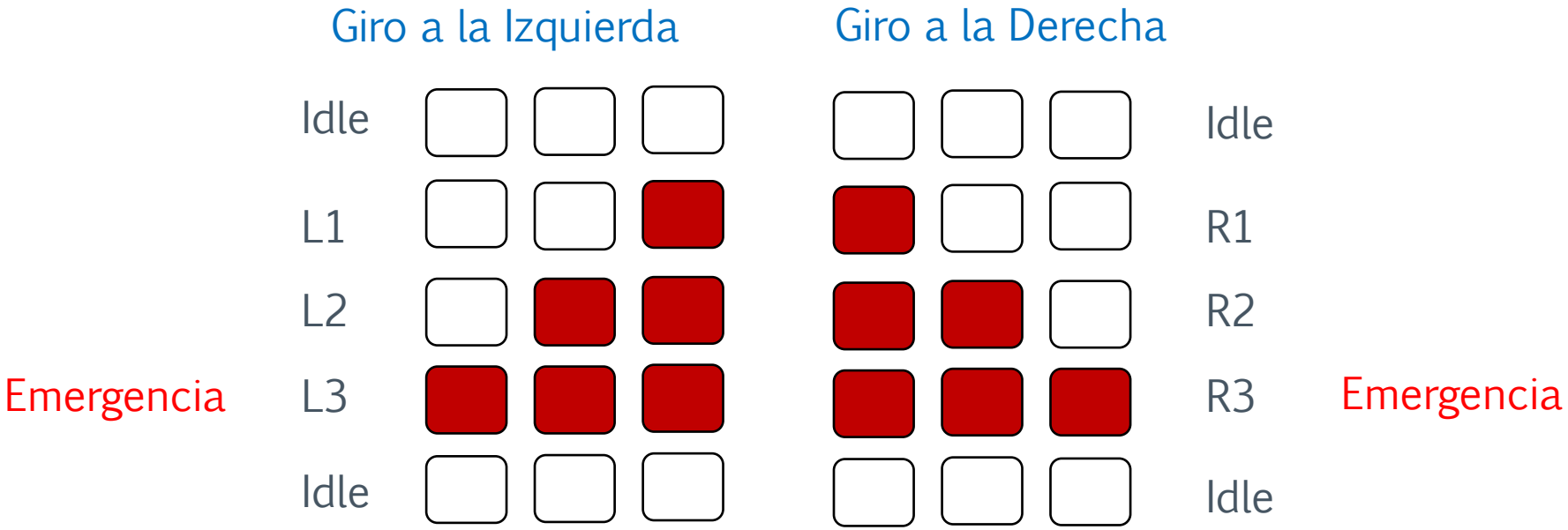




Un coleccionista tiene un vehículo Ford Thunderbird modelo 1965, como el que se indica en las figuras.

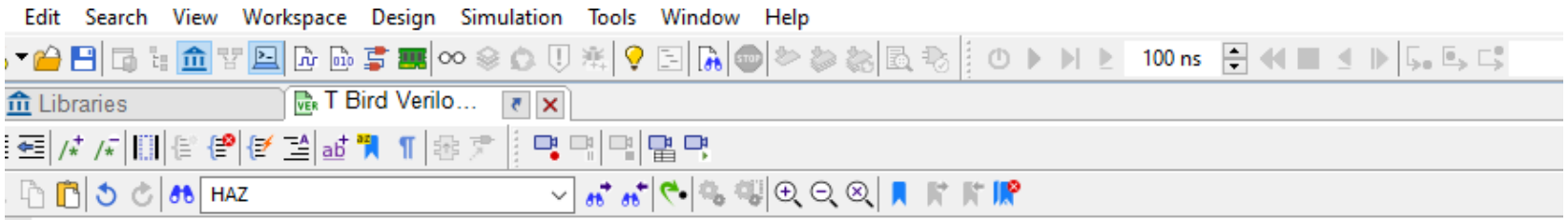
Ejercicio

Diseñe una máquina de estados en Verilog para controlar las luces de giro traseras del Thunderbird. Las mismas operan de la siguiente forma



El tablero del Ford tiene tres señales de control:

- Giro a la Izquierda
- Giro a la Derecha
- Balizas de Emergencia




```
module FordTbird ( CLOCK, RESET, IZQ, DER, EMER, LA, LB, LC, RA, RB, RC );
  input CLOCK, RESET, IZQ, DER, EMER;
  output reg LA, LB, LC, RA, RB, RC;
  reg [2:0] Sreg, Snext;                          // Registro de estado y de próximo estado
  parameter [2:0] IDLE = 3'b000,                  // Estado y Código de Estado
                  L1   = 3'b001,                  // Giro a la izquierda, una lámpara encendida
                  L2   = 3'b011,                  // Giro a la izquierda, dos lámparas encendidas
                  L3   = 3'b010,                  // Giro a la izquierda, tres lámparas encendidas
                  R1   = 3'b101,                  // Giro a la derecha, una lámpara encendida
                  R2   = 3'b111,                  // Giro a la derecha, dos lámparas encendidas
                  R3   = 3'b110,                  // Giro a la derecha, tres lámparas encendidas
                  LR3  = 3'b100;                  // Emergencia, todas las lámparas encendidas

  always @ (posedge CLOCK or posedge RESET)       // Carga de la memoria de estado
    if (RESET==1) Sreg <= IDLE; else Sreg <= Snext;     // Reset Asincrónico


  always @ (IZQ, DER, EMER, Sreg) begin           // Lógica del próximo estado
    case (Sreg)
      IDLE:   if (EMER | (IZQ & DER) )    Snext = LR3;
              else if (DER)               Snext = R1;
              else if (IZQ)               Snext = L1;
              else                        Snext = IDLE;
      R1:     if (EMER) Snext = LR3; else Snext = R2;
      R2:     if (EMER) Snext = LR3; else Snext = R3;
      R3:     if (EMER) Snext = LR3; else Snext = IDLE;
      L1:     if (EMER) Snext = LR3; else Snext = L2;
      L2:     if (EMER) Snext = LR3; else Snext = L3;
      L3:     if (EMER) Snext = LR3; else Snext = IDLE;
      LR3:    Snext = IDLE;
      default Snext = IDLE;
    endcase
  end
```

```
  always @ (Sreg) begin                                  // Lógica del próximo estado
    case (Sreg)
      IDLE:   {LC,LB,LA,RA,RB,RC} = 6'b000000;           // Todas las lámparas apagadas
      R1:     {LC,LB,LA,RA,RB,RC} = 6'b000100;           // Giro
      R2:     {LC,LB,LA,RA,RB,RC} = 6'b000110;           // a la
      R3:     {LC,LB,LA,RA,RB,RC} = 6'b000111;           // izquierda
      L1:     {LC,LB,LA,RA,RB,RC} = 6'b001000;           // Giro
      L2:     {LC,LB,LA,RA,RB,RC} = 6'b011000;           // a la
      L3:     {LC,LB,LA,RA,RB,RC} = 6'b111000;           // derecha
      LR3:    {LC,LB,LA,RA,RB,RC} = 6'b111111;           // emergencia
      default {LC,LB,LA,RA,RB,RC} = 6'b000000;           //
    endcase
  end
endmodule
```