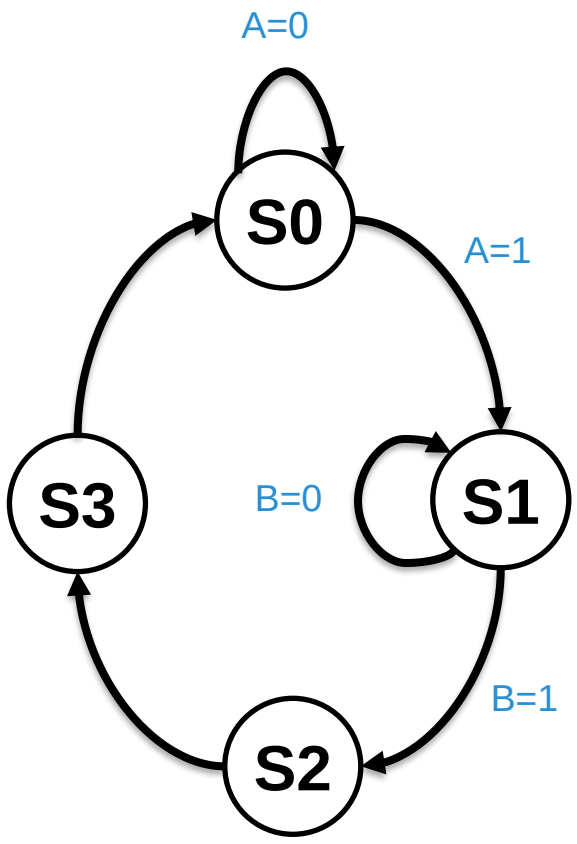
Máquina de Estados Finitos – Ej. Semáforos

Electrónica Digital 2 – 2023-2

Máquinas de Estado Finitos

Diagrama de estados

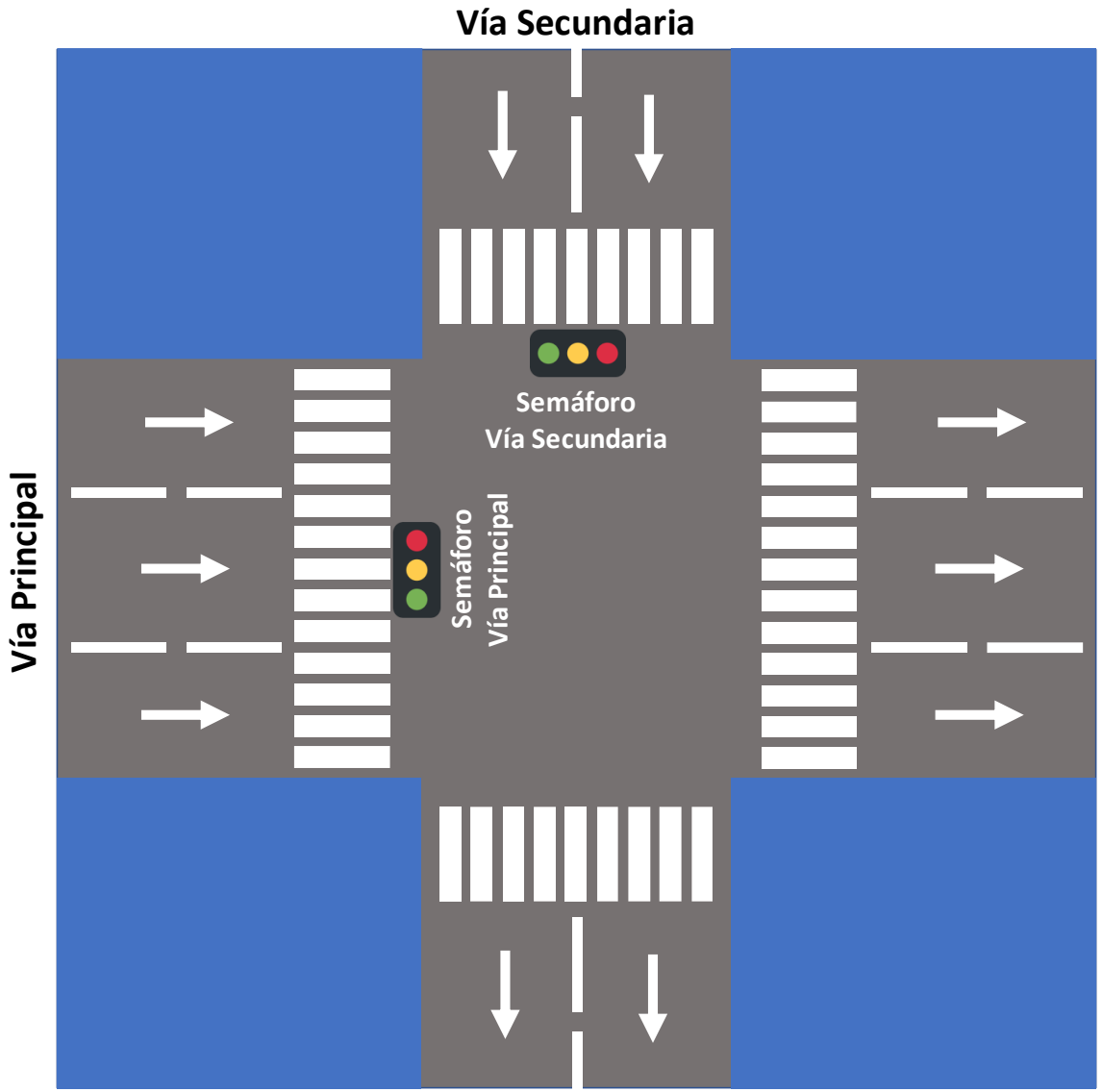


- Para cualquier estado, existen **finitos** estados próximos posibles
- En cada ciclo de reloj, la FSM cambia al siguiente estado
- En un diagrama, todas las posibles transiciones deben estar visibles
- Observe que las entradas y el estado actual determinan el estado siguiente
- El reloj, de cierta manera, es también una entrada
- El estado determina el valor de las salidas (Moore)

Máquinas de Estado Finitos (2)

Implementar una FSM para el controlador de semáforos

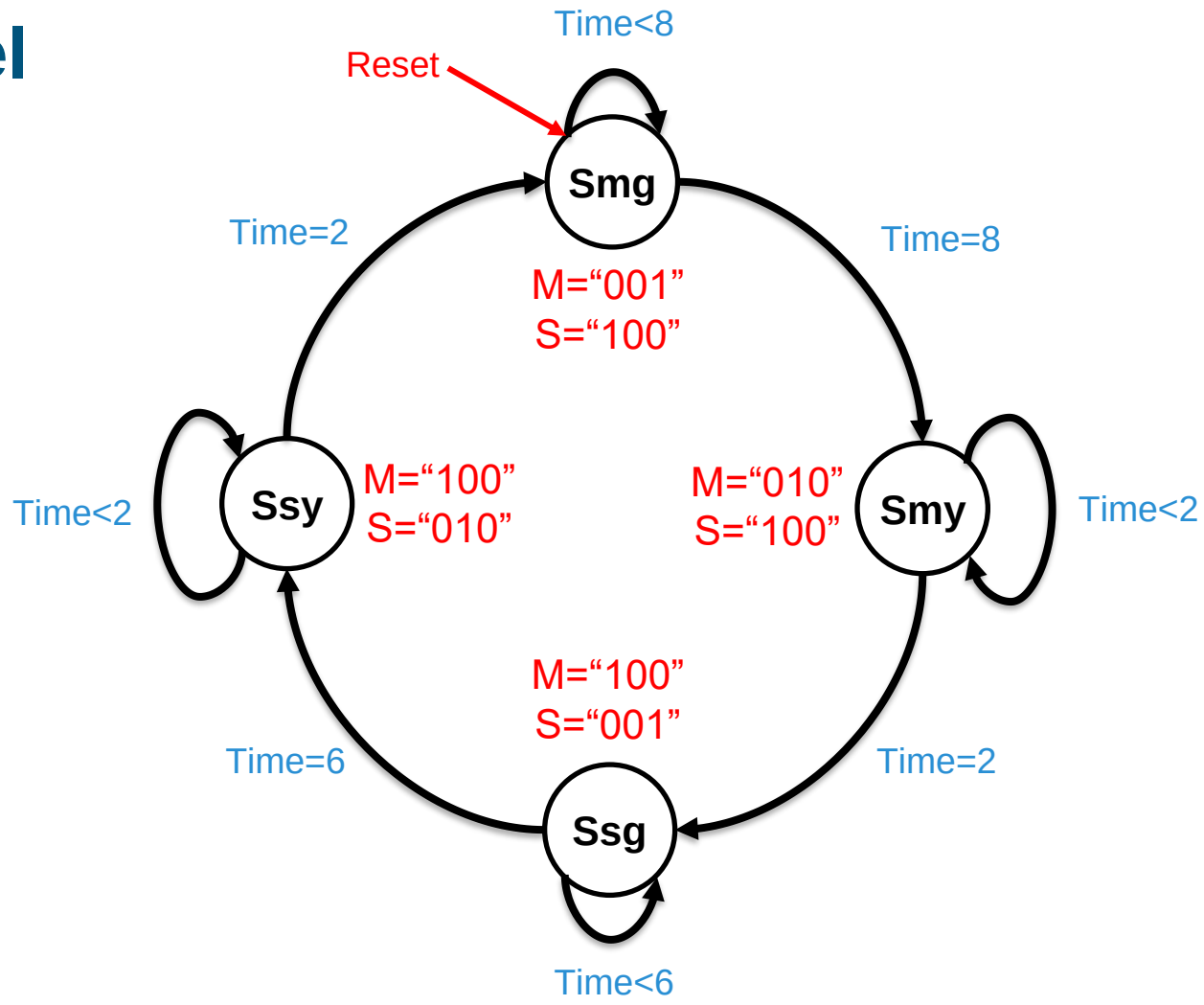
- **Vía principal**
  - **Verde:** 8 segundos
  - **Amarillo:** 2 segundos
- **Vía secundaria**
  - **Verde:** 6 segundos
  - **Amarillo:** 2 segundos



Máquinas de Estado Finitos (3)

Implementar una FSM para el controlador de semáforos

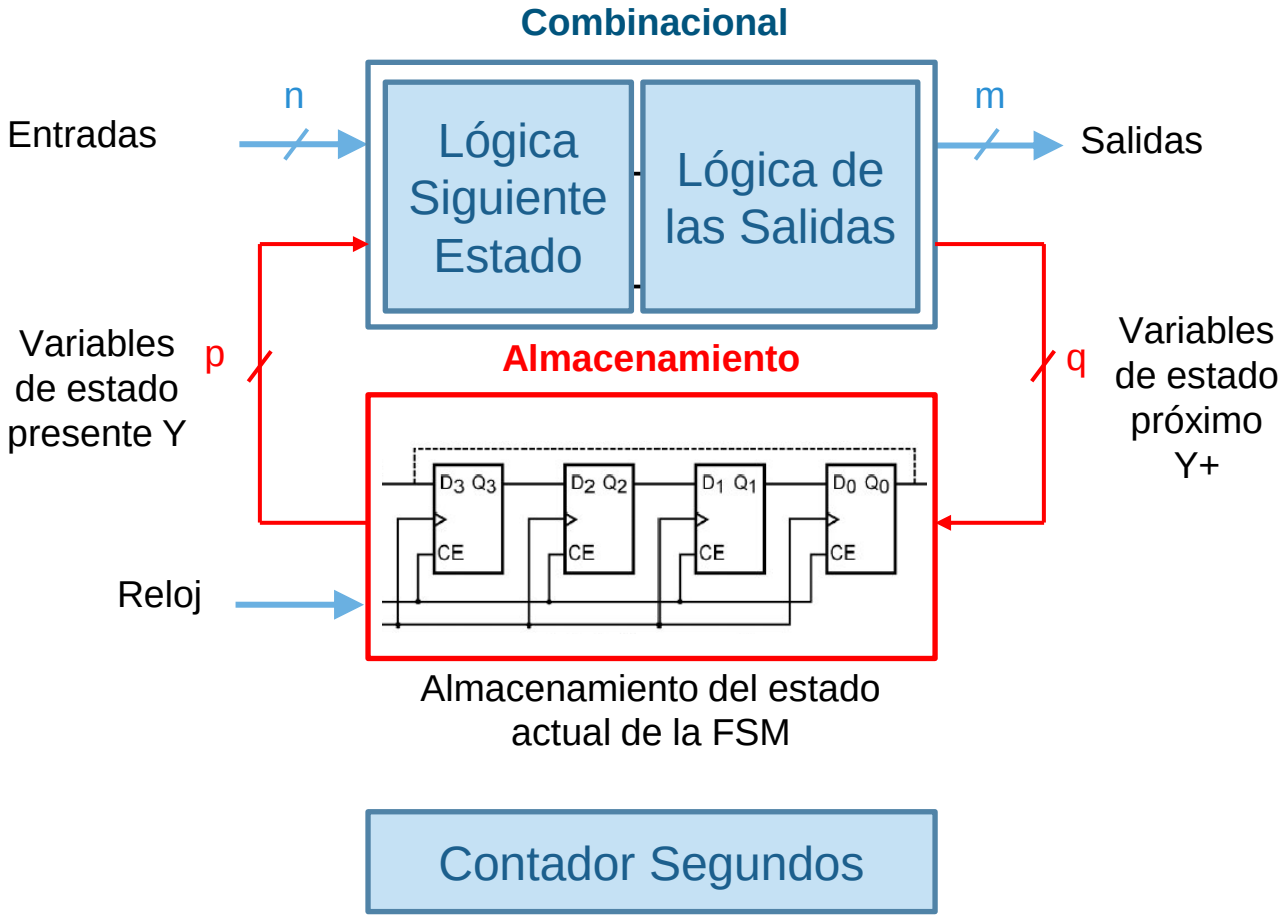
- **Vía principal**
  - **Verde:** 8 segundos
  - **Amarillo:** 2 segundos
- **Vía secundaria**
  - **Verde:** 6 segundos
  - **Amarillo:** 2 segundos



Máquinas de Estado Finitos (4)

Esquema para la implementación de la FSM

- Actualización estado actual
- Lógica del estado siguiente
- Lógica de las salidas
- Contador de segundos



Máquinas de Estado Finitos (5)

Definición de módulo, puertos entradas/salidas y señales

```
module trafficlight #(fpga_freq = 50_000_000,
    tgreen_main = 8, tyellow_main = 2, tgreen_sec = 6, tyellow_sec = 2)
    (clk, nreset, main_lights, sec_lights, clk_seconds);

    /* Entradas y salidas */
    input logic clk, nreset;
    output logic [2:0] main_lights;
    output logic [2:0] sec_lights;
    output logic clk_seconds;

    /* Circuito para invertir señal de reloj */
    logic reset;
    assign reset = ~nreset;
    /* Señales internas para contar segundos a partir del reloj de la FPGA */
    logic [3:0] cnt_seconds;    // 4-bits para contar hasta 16 segundos
    logic cnt_timeIsUp;         // Tiempo completado en estado actual
    cntdiv_n #(fpga_freq) cntDiv (clk, reset, clk_seconds);
    ...
endmodule
```

Máquinas de Estado Finitos (6)

Actualización estado actual de la FSM

```
module trafficlight #(fpga_freq = 50_000_000,
    tgreen_main = 8, tyellow_main = 2, tgreen_sec = 6, tyellow_sec = 2)
    (clk, nreset, main_lights, sec_lights, clk_seconds);
    ...
    /* Estados de la FSM y señales internas para estado actual y siguiente */
    typedef enum logic [1:0] {Smg, Smy, Ssg, Ssy} State;
    State currentState, nextState;

    /* Circuito secuencial para actualizar estado actual con el siguiente */
    always_ff @(posedge clk_seconds, posedge reset)
       if (reset)
          currentState <= Smg;
       else
          currentState <= nextState;
    ...
endmodule
```

Máquinas de Estado Finitos (7)

Lógica del estado siguiente

```
module trafficlight #(fpga_freq = 50_000_000,
   ...
   /* Circuito combinacional para determinar siguiente estado de la FSM */
   always_comb begin
      nextState = currentState;    // Para evitar escribir los else
      case (currentState)
         Smg:
            if(cnt_timeIsUp)
               nextState = Smy;
         Smy:
            if(cnt_timeIsUp)
               nextState = Ssg;
         Ssg:
            if(cnt_timeIsUp)
               nextState = Ssy;
         Ssy:
            if(cnt_timeIsUp)
   ...
endmodule
```

Máquinas de Estado Finitos (8)

Lógica de las salidas

```
module trafficlight #(fpga_freq = 50_000_000,
   ...
   /* Circuito combinacional para manejar las salidas */
   always_comb begin
      main_lights = 3'b100;    // Para simplificar cada case
      sec_lights = 3'b100;     // Para simplificar cada case
      case (currentState)
         Smg:
            main_lights = 3'b001;
         Smy:
            main_lights = 3'b010;
         Ssg:
            sec_lights = 3'b001;
         Ssy:
            sec_lights = 3'b010;
      endcase
   end
   ...
endmodule
```

Máquinas de Estado Finitos (9)

Contador de segundos

```
module trafficlight #(fpga_freq = 50_000_000,
   ...
   /* Circuito secuencial para el contador de segundos */
   always_ff @(posedge clk_seconds, posedge reset) begin
      if (reset) begin
         cnt_seconds <= 0;
         cnt_timeIsUp <= 0;
      end else begin
         cnt_seconds <= cnt_seconds + 1; // No cambia hasta finalizar proc. (<=)
         cnt_timeIsUp <= 0;
         case (currentState)
            Smg:
               if(cnt_seconds == (tgreen_main-1)) begin
                  cnt_timeIsUp <= 1;
                  cnt_seconds <= 0;
               end
            Smy:
   ...
endmodule
```

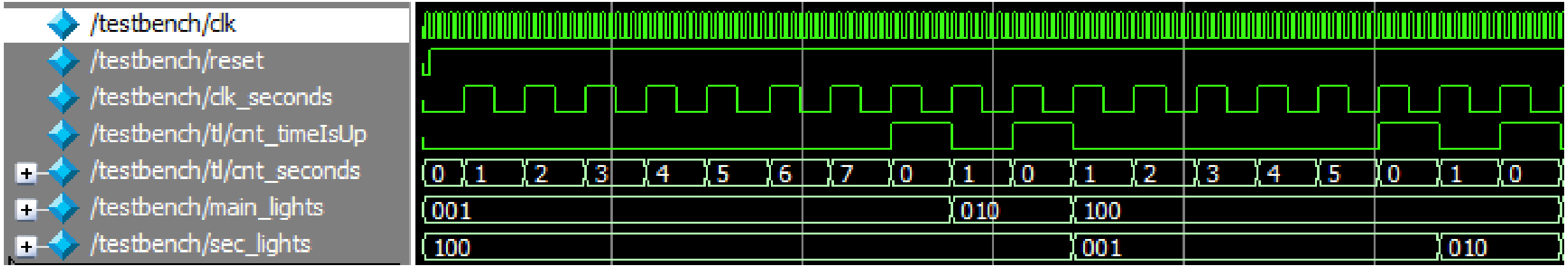
Máquinas de Estado Finitos (10)

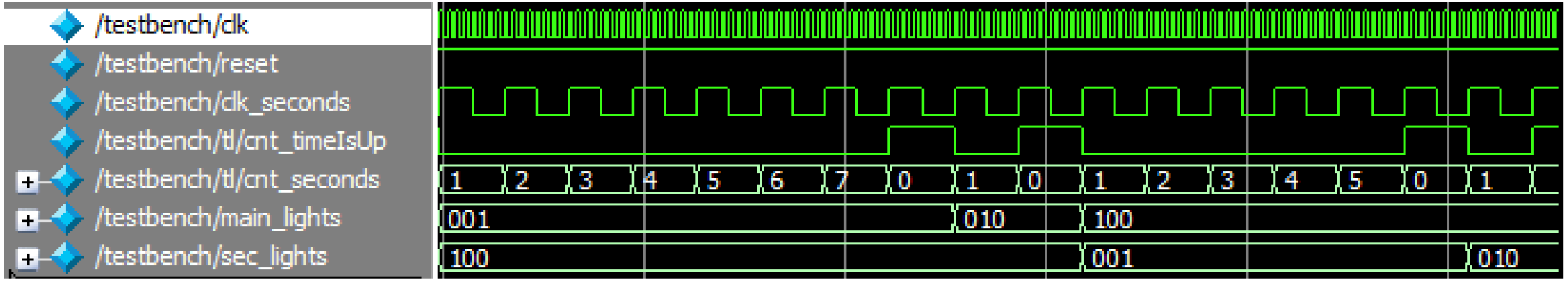
Divisor de Frecuencia

```
module cntdiv_n #(TOPVALUE = 10) (clk, rst, clkdiv);
   input logic clk, rst;
   output logic clkdiv;
   // Bits for the counter
   localparam BITS = $clog2(TOPVALUE);
   // counter register
   logic [BITS - 1 : 0] rCounter;
   // increment or reset the counter
   always @(posedge clk, posedge rst) begin
      if (rst) begin
         rCounter <= 0;
         clkdiv <= 0;
      end else begin
         if (rCounter == (TOPVALUE - 1))
            rCounter <= 0;
         else
            rCounter <= rCounter + 1;
         // Guarantee a registered clock
         clkdiv <= (rCounter >= (TOPVALUE/2)) ? 1'b1 : 1'b0;
      end
   end
endmodule
```

Máquinas de Estado Finitos (11)

Simulación





Electrónica Digital 2 – 2023-2

Fin del tema: Máquina de Estados Finitos – Ej. Semáforos