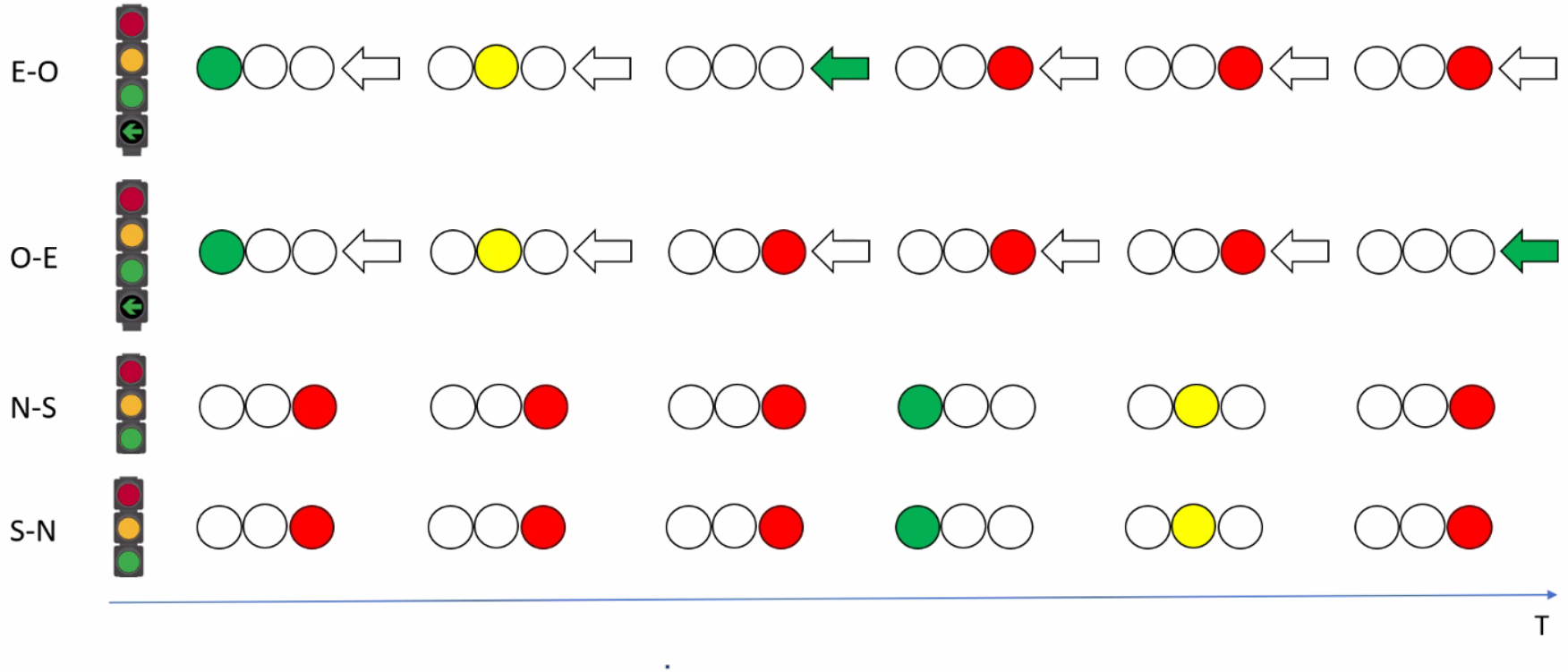
Actividad 3.1 Semáforo

J. Emilio Soriano Campos

Este problema consistió en construir un autómata en VHDL basándonos en una máquina de estados de Moore que representara el funcionamiento de un semáforo. Además el semáforo tenia que tenerla posibilidad de mostrar diferentes tiempos para los semáforos. El orden de los estados es el siguiente:

Análisis del problema

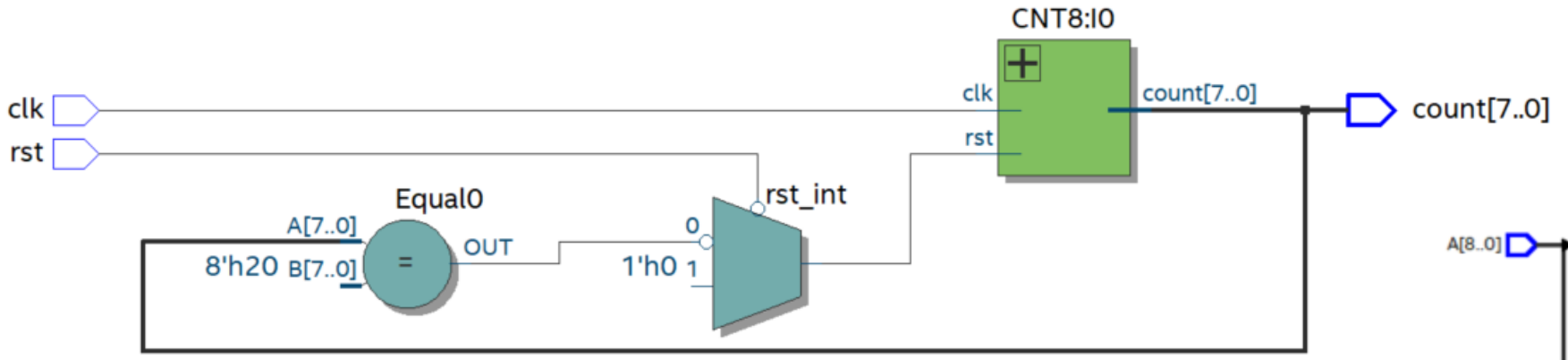


Mi solución consistió en crear un contador modulo 32 para dividir los 6 estados entre los 32 espacios que me brinda el contador. Para el modulo 32 necesite construir primero un sumador de 8 bits ya que 32 representado en binario tiene 6 bits de tamaño. Despues tuve que construir un contador de 8 bits y finalmente usar ambos para construir el modulo 32.

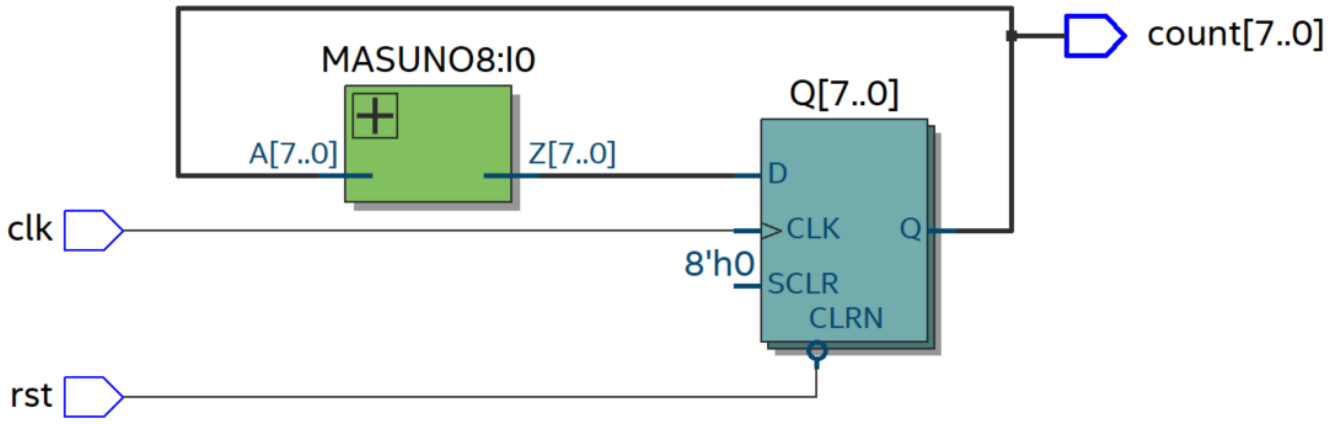
```vhdl
-- J. Emilio Soriano Campos
-- Sumador de 8 bits
library ieee;
use ieee.std_logic_1164.all;

entity MASUNO8 is
   port (A : in std_logic_vector(7 downto 0);
         Z : out std_logic_vector(7 downto 0));
end entity;

architecture MASUNO8_ARC of MASUNO8 is
   component HA is
   port (A, B : in std_logic;
         s, Co : out std_logic );
   end component;

   signal C : std_logic_vector(7 downto 1);

   begin
   I0 : HA port map (A(0), '1', Z(0), C(1));
   I1 : HA port map (A(1), C(1), Z(1), C(2));
   I3 : HA port map (A(2), C(2), Z(2), C(3));
   I4 : HA port map (A(3), C(3), Z(3), C(4));
   I5 : HA port map (A(4), C(4), Z(4), C(5));
   I6 : HA port map (A(5), C(5), Z(5), C(6));
   I7 : HA port map (A(6), C(6), Z(6), C(7));

   z(7) <= A(7) xor C(7);
end architecture;
```

```vhdl
-- J. Emilio Soriano Campos
-- Contador de 8 bits
library ieee;
use ieee.std_logic_1164.all;

entity CNT8 is
   port (clk, rst : in std_logic;
         count : out std_logic_vector(7 downto 0));
end entity;

architecture CNT8_ARC of CNT8 is
   component MASUNO8 is
   port (A : in std_logic_vector(7 downto 0);
         Z : out std_logic_vector(7 downto 0));
   end component;

   signal D, Q : std_logic_vector(7 downto 0);

   begin
   I0 : MASUNO8 port map (Q, D);
   count <= Q;

   P1 : process(clk, rst)
      begin
      if rst = '0' then
         Q <= "00000000";
      elsif clk'event and clk = '1' then
         Q <= D;
      end if;
   end process;
end architecture;
```

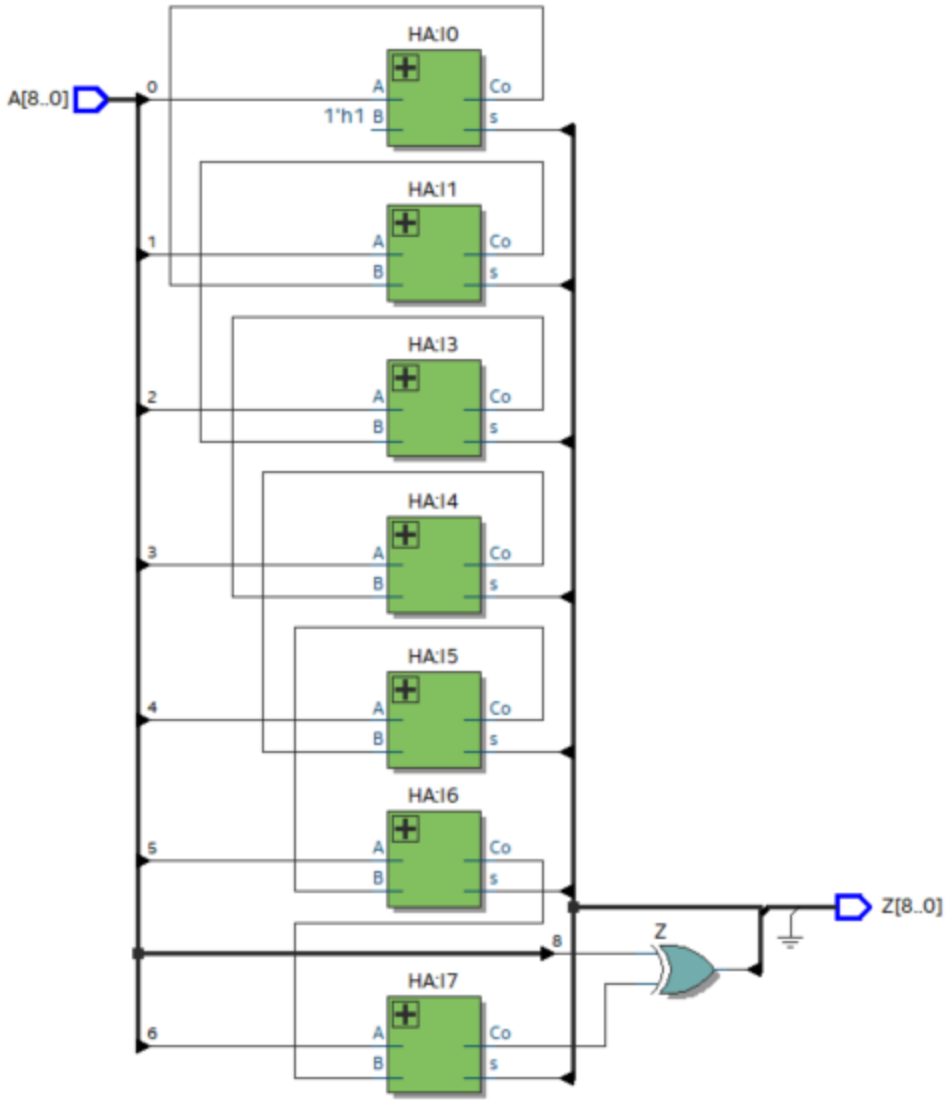
```vhdl
-- J. Emilio Soriano Campos
-- Contador de modulo 32
library ieee;
use ieee.std_logic_1164.all;

entity MOD32 is
   port (clk, rst : in std_logic;
         count : out std_logic_vector(7 downto 0));
end entity;

architecture MOD32_ARC of MOD32 is
   component CNT8 is
   port (clk, rst : in std_logic;
         count : out std_logic_vector(7 downto 0));
   end component;

   signal rst_int : std_logic;
   signal Q : std_logic_vector(7 downto 0);

   begin
   I0 : CNT8 port map (clk, rst_int, Q);

   P1 : process (rst, Q)
      begin
      case rst is
         when '0' => rst_int <= '0';
         when others => if Q = "00100000" then
                           rst_int <= '0';
                        else
                           rst_int <= '1';
                        end if;
      end case;
   end process;
   count <= Q;
end architecture;
```



Mod 32



Contador de 8 bits



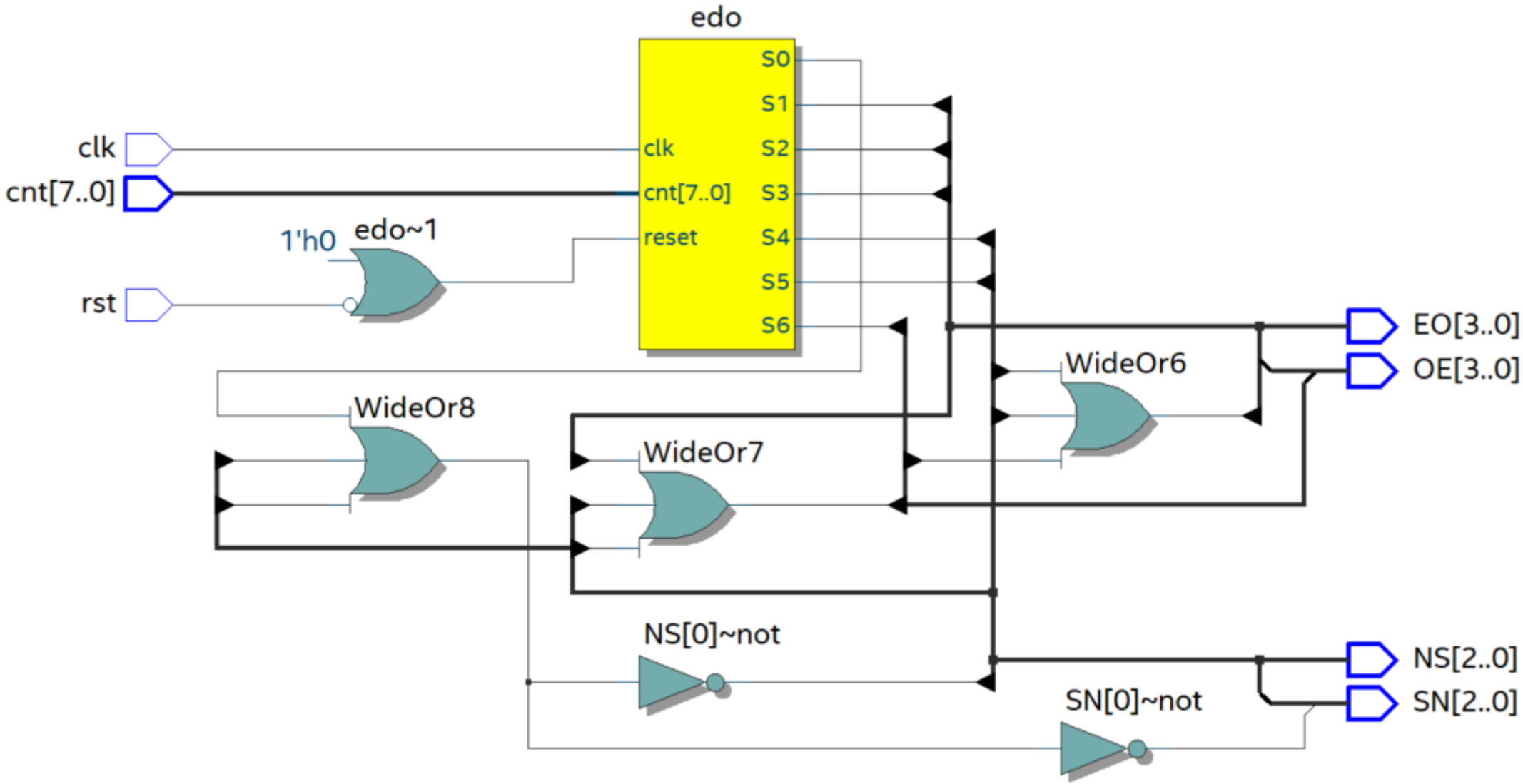
Sumador de 8 bits

Después de haber construido el módulo 32 ahora podía empezar a construir la maquina de estados de tipo Moore para definir los diferentes estados para las “luces” del semáforo y así fue como quedo:

```vhdl
-- J. Emilio Soriano Campos
-- Maquina de estados de tipo Moore para semaforo
 library ieee;
 use ieee.std_logic_1164.all;

entity semaforo is
   port (clk, rst : in std_logic;
         cnt : in std_logic_vector(7 downto 0);
         EO, OE : out std_logic_vector(3 downto 0);
         NS, SN : out std_logic_vector(2 downto 0));
end entity;

architecture semaforo_ARC of semaforo is
   component MOD32 is
   port (clk, rst : in std_logic;
         count : out std_logic_vector(7 downto 0));
   end component;

   type ESTADOS is (S0, S1, S2, S3, S4, S5, S6);
   signal edo, edof : ESTADOS;

   begin
   p1 : process (clk, rst)
      begin
      if rst = '0' then
         edo <= S0;
      elsif clk'event and clk = '1' then
         edo <= edof;
      end if;
   end process;
```

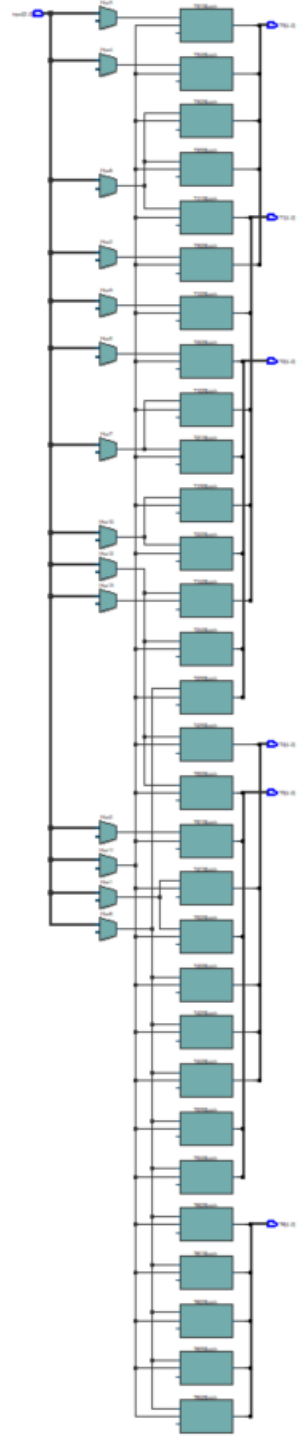
```vhdl
   p2 : process (edo, cnt)
      begin
      case edo is
         when S0 => if cnt = "00000000" then
                       edof <= S0;
                    else
                       edof <= S1;
                    end if;
         when S1 => if cnt > "00000000" and cnt < "00000111" then
                       edof <= S1;
                    else
                       edof <= S2;
                    end if;
         when S2 => if cnt > "00001000" and cnt < "00001011" then
                       edof <= S2;
                    else
                       edof <= S3;
                    end if;
         when S3 => if cnt > "00001100" and cnt < "00001111" then
                       edof <= S3;
                    else
                       edof <= S4;
                    end if;
         when S4 => if cnt > "00010000" and cnt < "00010111" then
                       edof <= S4;
                    else
                       edof <= S5;
                    end if;
         when S5 => if cnt > "00011000" and cnt < "00011011" then
                       edof <= S5;
                    else
                       edof <= S6;
                    end if;
         when S6 => if cnt > "00011100" and cnt < "00011111" then
                       edof <= S6;
                    else
                       edof <= S1;
                    end if;
         when others => null;
      end case;
   end process;
```

```vhdl
   p3 : process (edo)
      begin
      case edo is
         when S0 => EO <= "0000";
                    OE <= "0000";
                    NS <= "000";
                    SN <= "000";

         when S1 => EO <= "1000";
                    OE <= "1000";
                    NS <= "001";
                    SN <= "001";

         when S2 => EO <= "0100";
                    OE <= "0100";
                    NS <= "001";
                    SN <= "001";

         when S3 => EO <= "0001";
                    OE <= "0010";
                    NS <= "001";
                    SN <= "001";

         when S4 => EO <= "0010";
                    OE <= "0010";
                    NS <= "100";
                    SN <= "100";

         when S5 => EO <= "0010";
                    OE <= "0010";
                    NS <= "010";
                    SN <= "010";

         when S6 => EO <= "0010";
                    OE <= "0001";
                    NS <= "001";
                    SN <= "001";
         when others => null;
      end case;
   end process;
end architecture;
```


edo
S0
S1
S2
S3
S4
S5
S6
clk
cnt[7..0]
reset
clk
cnt[7..0]
1'h0
edo~1
rst
WideOr8
WideOr7
WideOr6
NS[0]~not
SN[0]~not
EO[3..0]
OE[3..0]
NS[2..0]
SN[2..0]

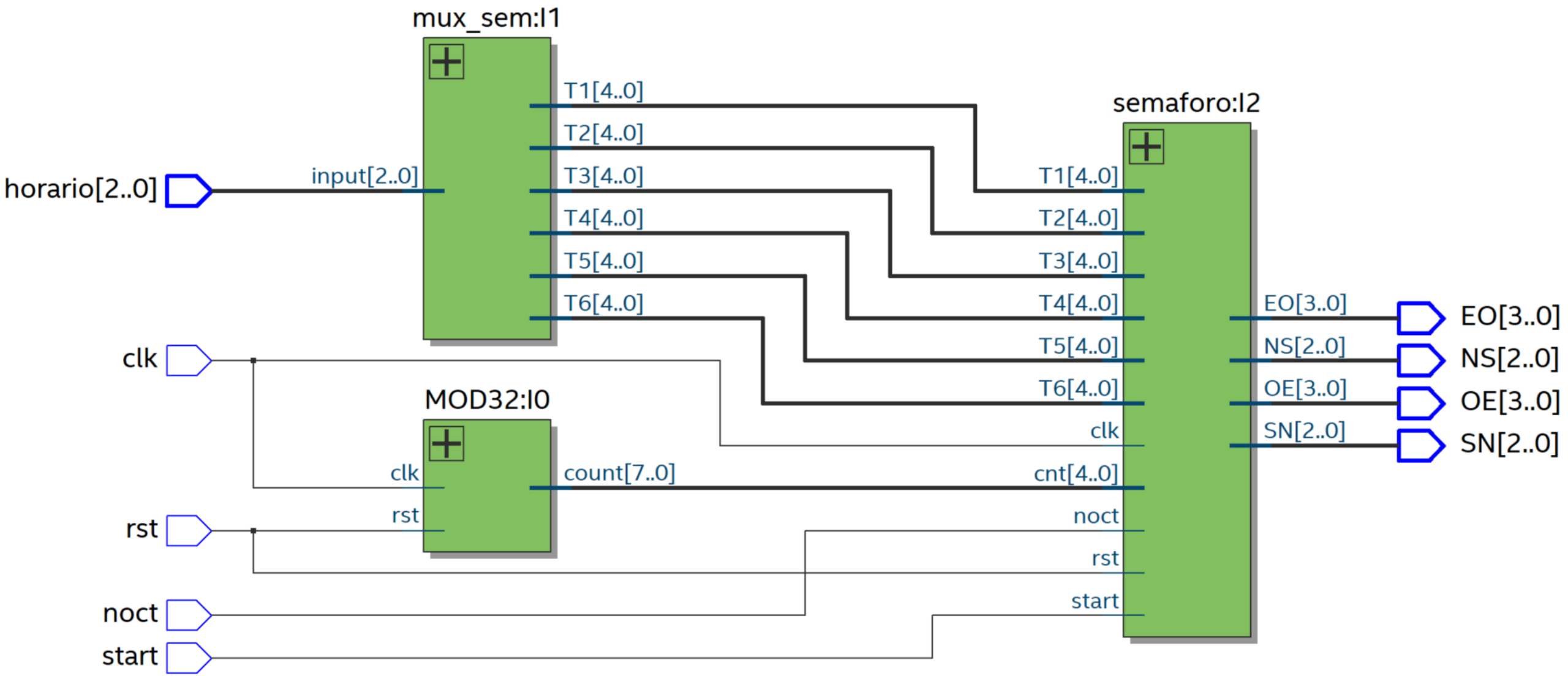
Para definir el tiempo que tenia que tener cada estado dependiendo del “tiempo del día” cree un multiplexor que recibía 3 bits y regresaba 6 variables a las que se les asignaba un valor dependiendo de la combinación de los 3 bits de entrada

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity mux_sem is
5     port (input : in std_logic_vector(2 downto 0);
6           T1, T2, T3, T4, T5, T6 : out std_logic_vector(4 downto 0));
7  end entity;
8
9  architecture mux_sem_ARC of mux_sem is
10    begin
11    p1 : process(input)
12       begin
13       case input is
14          when "001" => T1 <= "00101"; T2 <= "01010"; T3 <= "10000"; T4 <= "10101"; T5 <= "11010"; T6 <= "11111";
15          when "010" => T1 <= "01001"; T2 <= "01110"; T3 <= "10011"; T4 <= "10101"; T5 <= "11010"; T6 <= "11111";
16          when "011" => T1 <= "01000"; T2 <= "01101"; T3 <= "10001"; T4 <= "10101"; T5 <= "11010"; T6 <= "11111";
17          when "100" => T1 <= "00111"; T2 <= "01011"; T3 <= "01111"; T4 <= "10101"; T5 <= "11010"; T6 <= "11111";
18          when "101" => T1 <= "00111"; T2 <= "01010"; T3 <= "01101"; T4 <= "10111"; T5 <= "11100"; T6 <= "11111";
19          when others => null;
20       end case;
21    end process;
22 end architecture;
```



Con esto ya solo tenia que conectar el multiplexor y el contador modulo 32 a la maquina de estados para que funcionara por su cuenta:

```
1  -- J. Emilio Soriano Campos
2  -- Semaforo automata
3   library ieee;
4   use ieee.std_logic_1164.all;
5
6  entity semaforo_final is
7      port (clk, rst, start, noct : in std_logic;
8            horario : in std_logic_vector(2 downto 0);
9            EO, OE : out std_logic_vector(3 downto 0);
10           NS, SN : out std_logic_vector(2 downto 0));
11  end entity;
12
13 architecture semaforo_final_ARC of semaforo_final is
14     component MOD32 is
15     port (clk, rst : in std_logic;
16           count : out std_logic_vector(7 downto 0));
17     end component;
18     component mux_sem is
19     port (input : in std_logic_vector(2 downto 0);
20           T1, T2, T3, T4, T5, T6 : out std_logic_vector(4 downto 0));
21     end component;
22     component semaforo is
23     port (clk, rst, start, noct : in std_logic;
24           T1, T2, T3, T4, T5, T6 : in std_logic_vector(4 downto 0);
25           cnt : in std_logic_vector(4 downto 0);
26           EO, OE : out std_logic_vector(3 downto 0);
27           NS, SN : out std_logic_vector(2 downto 0));
28     end component;
29
30     signal contador : std_logic_vector(7 downto 0);
31     signal TI1, TI2, TI3, TI4, TI5, TI6 : std_logic_vector(4 downto 0);
32
33     begin
34     I0 : MOD32 port map (clk, rst, contador);
35     I1 : mux_sem port map (horario, TI1, TI2, TI3, TI4, TI5, TI6);
36     I2 : semaforo port map (clk, rst, start, noct, TI1, TI2, TI3, TI4, TI5, TI6, contador(4 downto 0), EO, OE, NS, SN);
37  end architecture;
```


mux_sem:I1
input[2..0]
T1[4..0]
T2[4..0]
T3[4..0]
T4[4..0]
T5[4..0]
T6[4..0]
horario[2..0]
clk
MOD32:I0
clk
rst
count[7..0]
rst
noct
start
semaforo:I2
T1[4..0]
T2[4..0]
T3[4..0]
T4[4..0]
T5[4..0]
T6[4..0]
clk
cnt[4..0]
noct
rst
start
EO[3..0]
NS[2..0]
OE[3..0]
SN[2..0]
EO[3..0]
NS[2..0]
OE[3..0]
SN[2..0]

La simulación quedo de esta manera: