

Cómo utilizar el editor de máquinas de estados de Quartus

Creador: David Rubio G.

Entrada: <https://soceame.wordpress.com/2025/01/15/como-utilizar-el-editor-de-maquinas-de-estados-de-quartus/>

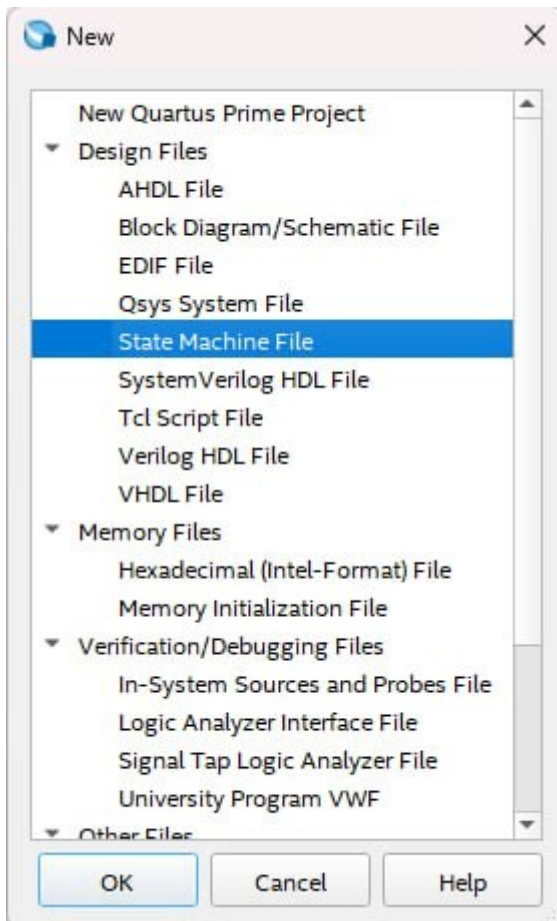
Blog: <https://soceame.wordpress.com/>

GitHub: <https://github.com/DRubioG>

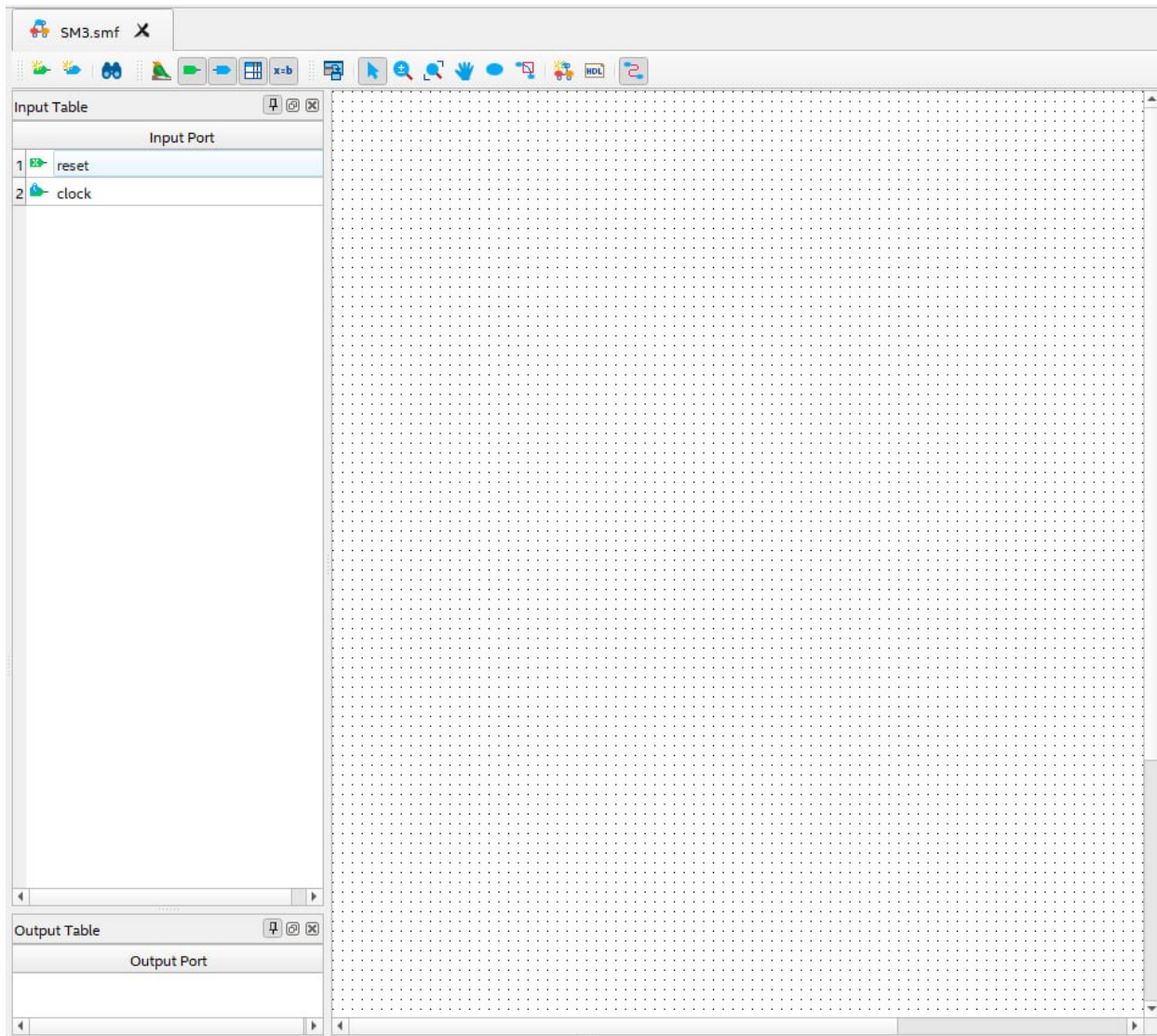
Fecha última modificación: 24/02/2025

Dentro de Quartus existe un editor de máquinas de estados de forma interactiva. Para acceder a él tenemos que darle a crear un nuevo archivo.

En la opción de crear archivo tenemos que crear un *State Machine File*.



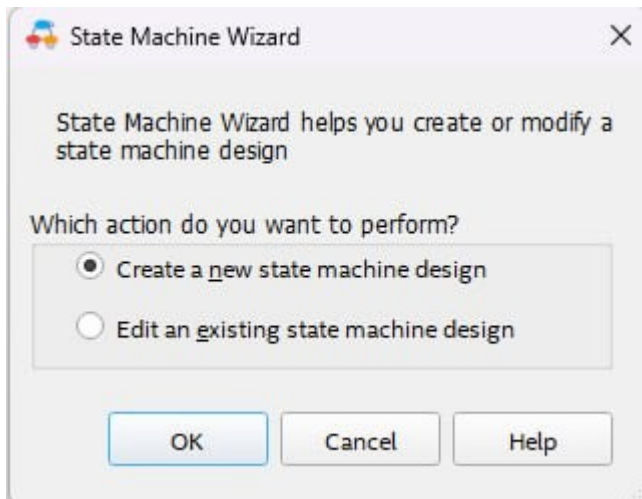
Esto abre el editor de máquinas de estados.



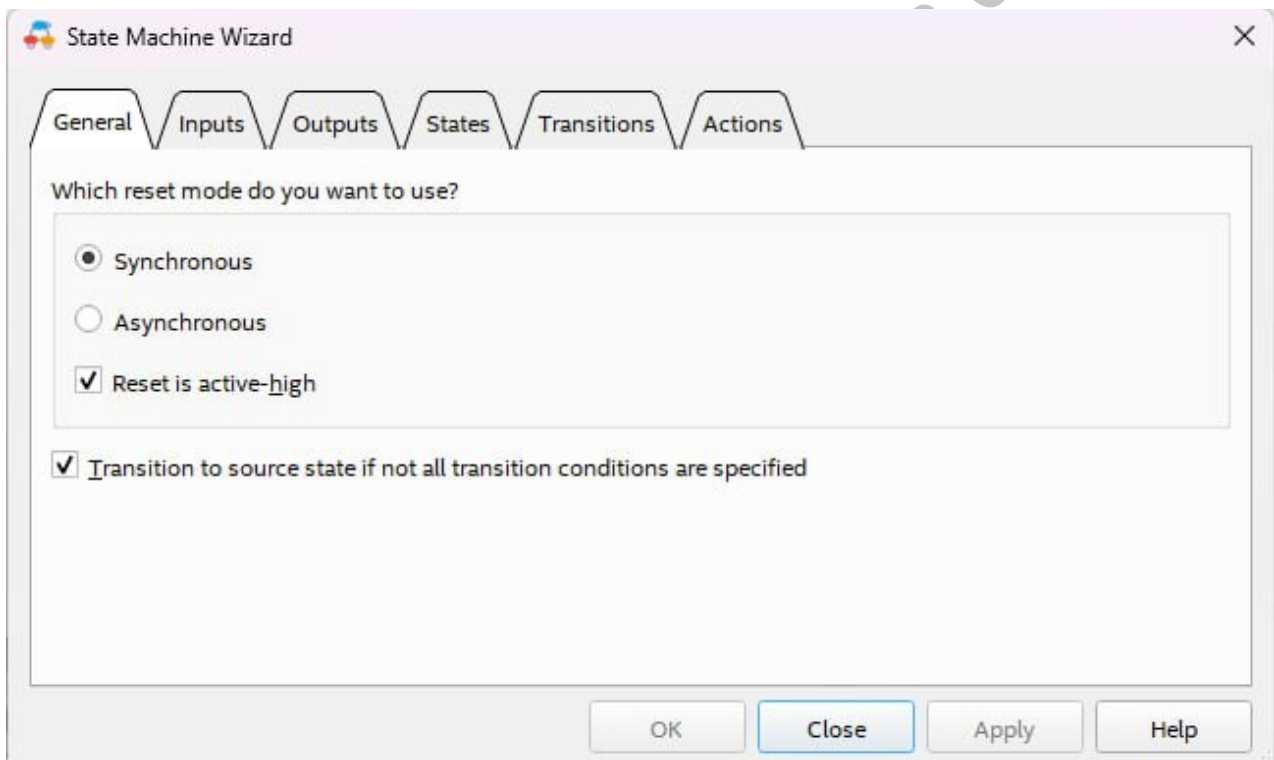
Ahora le damos al botón de colores.



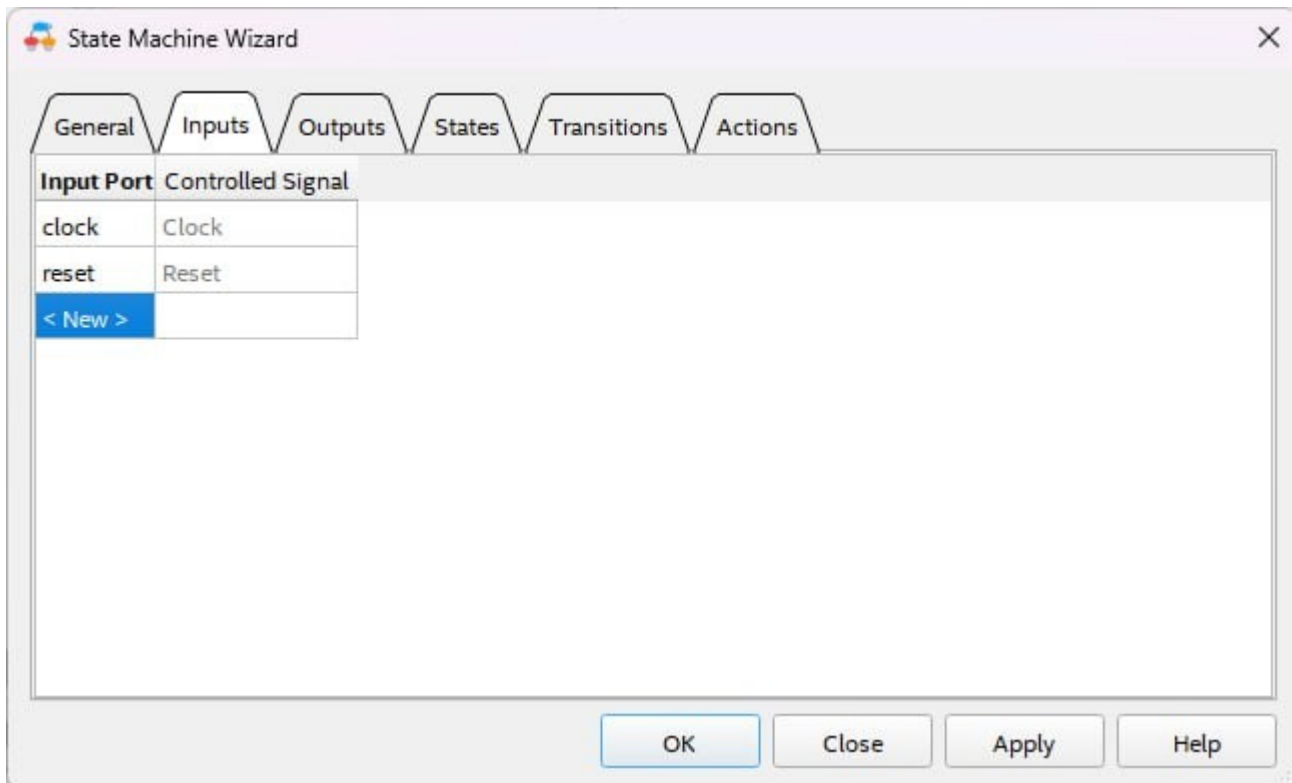
Y nos pregunta si queremos modificar una máquina de estados nueva o editar una antigua.



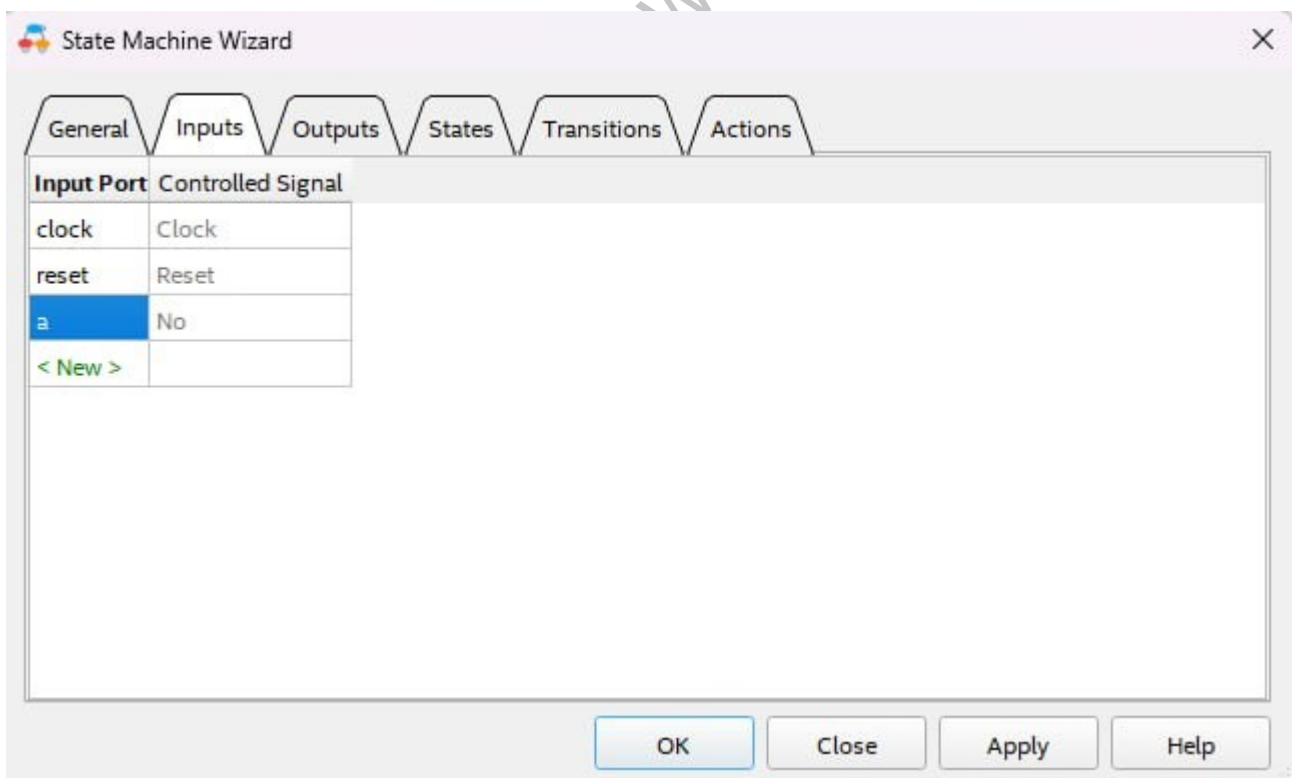
Lo primero que nos pregunta si queremos una máquina de estados síncrona o asíncrona.



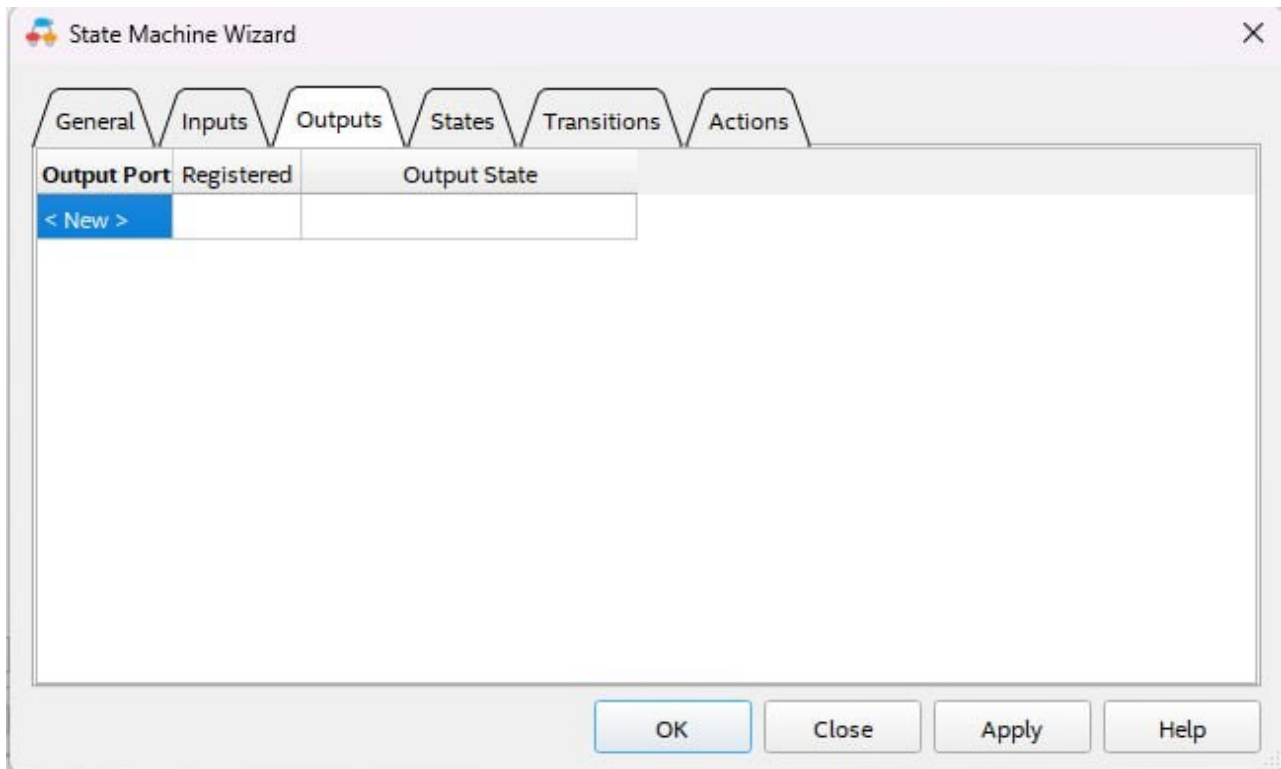
En la siguiente pestaña nos pregunta por las entradas a la máquina de estados.



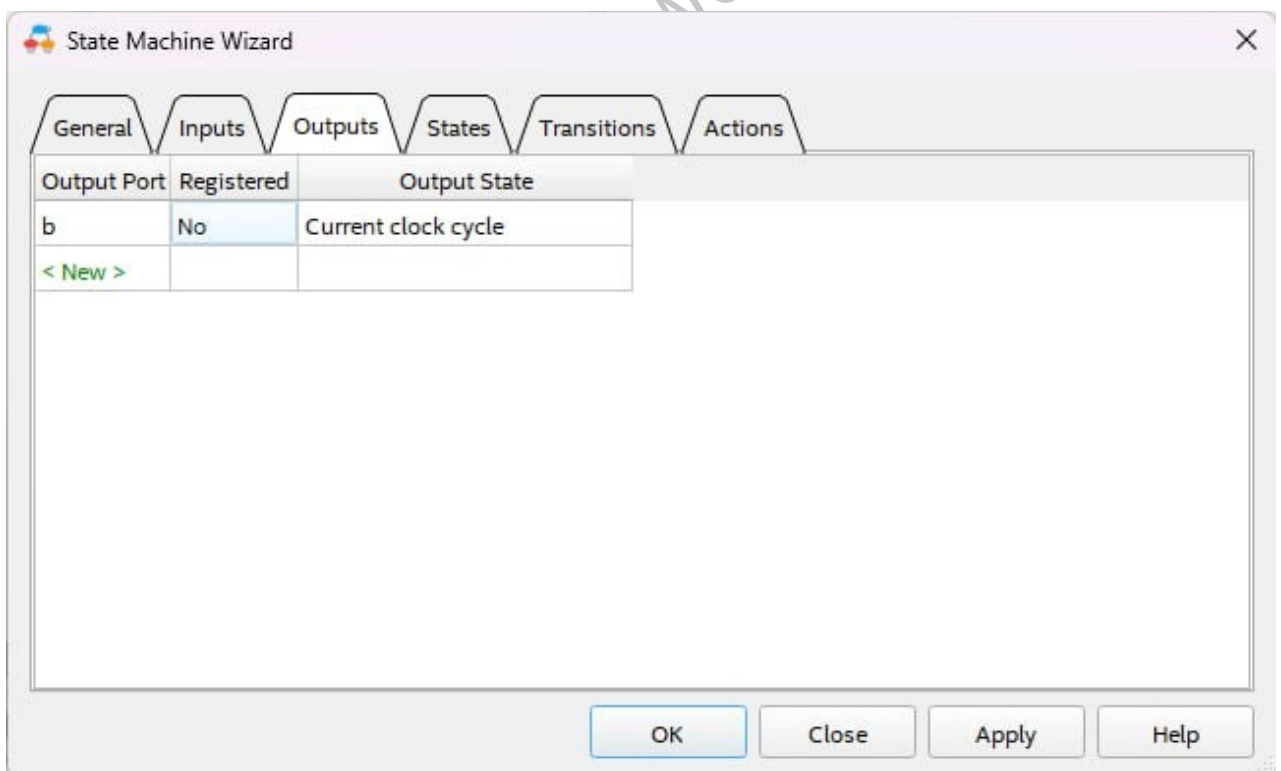
Aquí hay que introducir todas la entradas que generan cambios de estado en nuestra máquina.



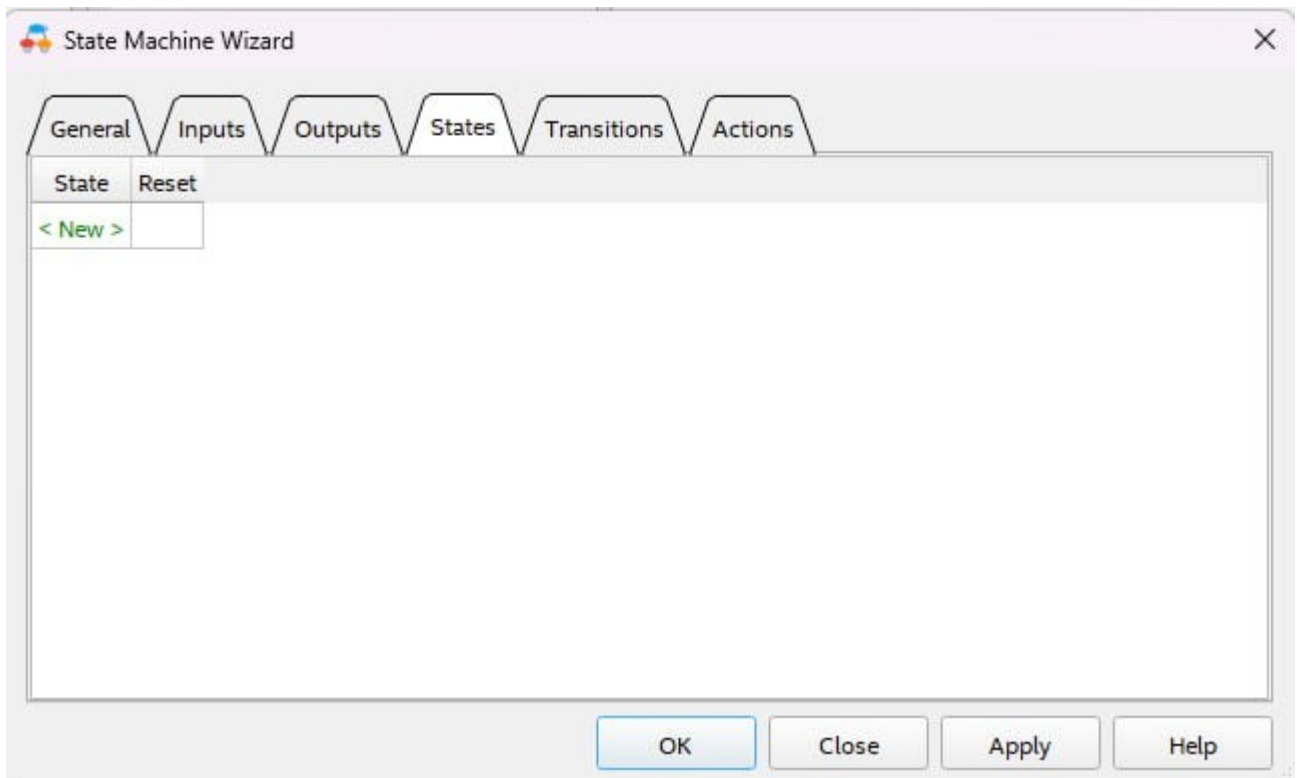
La siguiente pestaña nos pregunta por las salidas de nuestra máquina de estados.



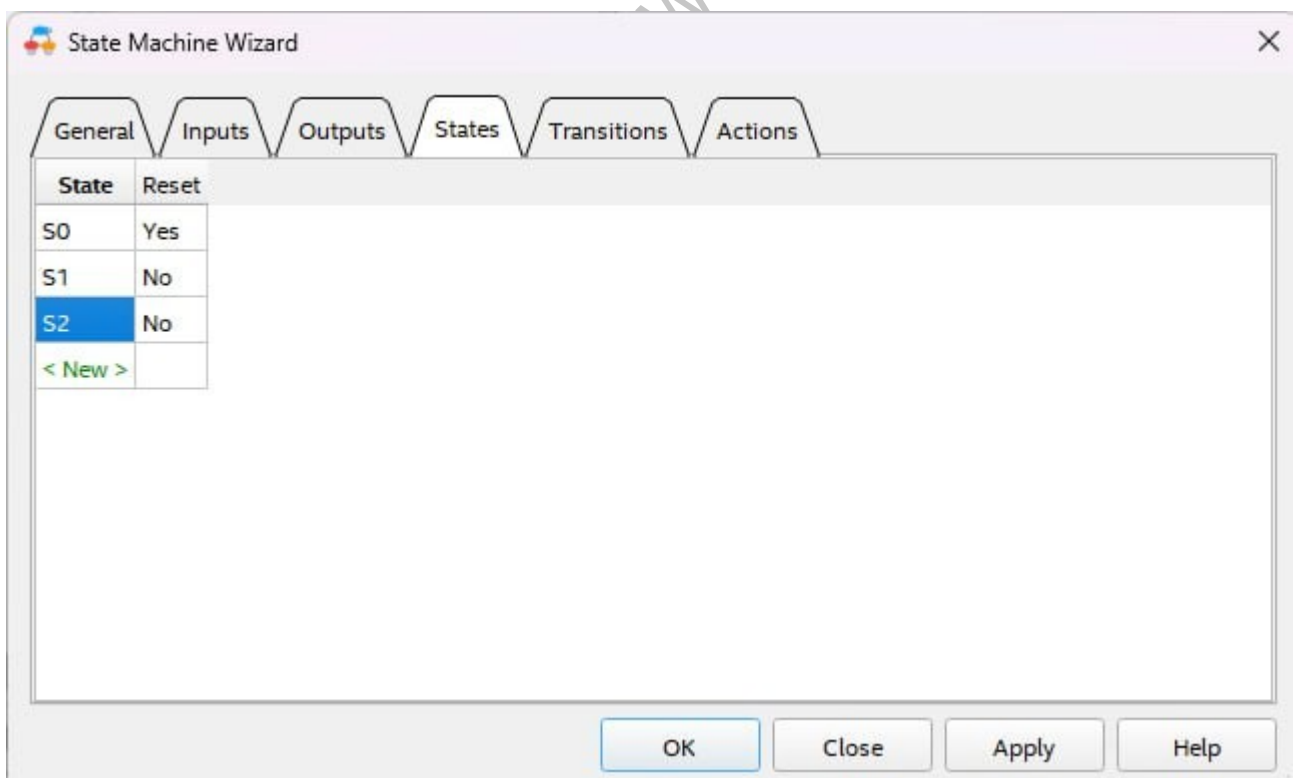
Ahora le introducimos las salidas que queremos tener.



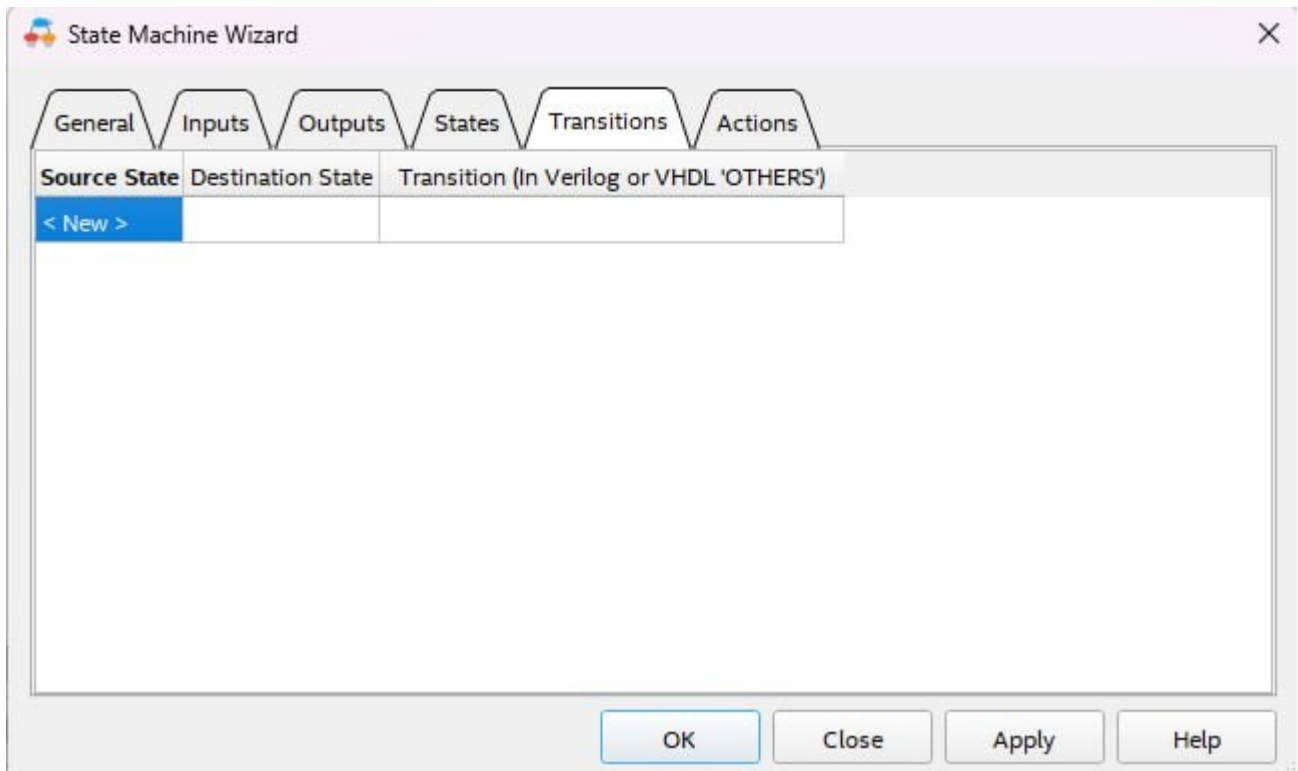
Lo siguiente que nos pide son los estados de nuestra máquina.



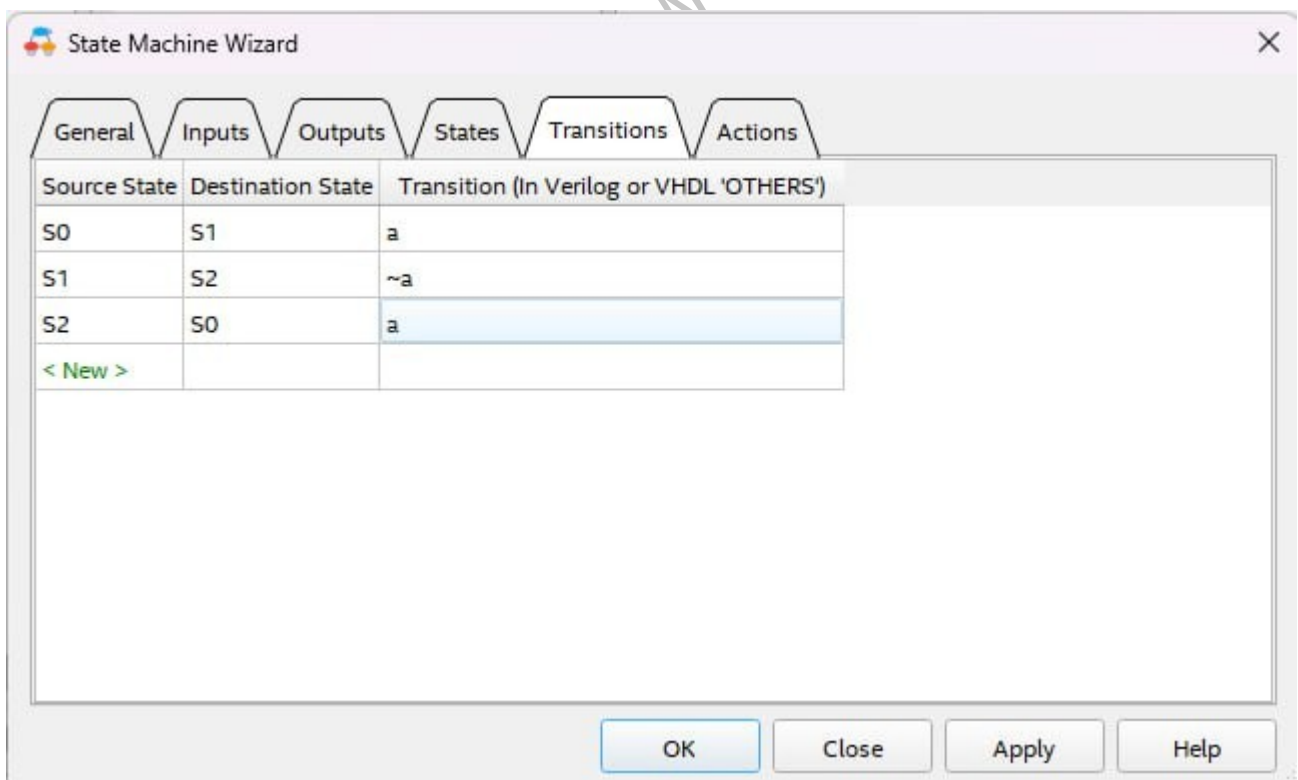
Al introducir los estados nos pregunta si alguno tiene señal de reset.



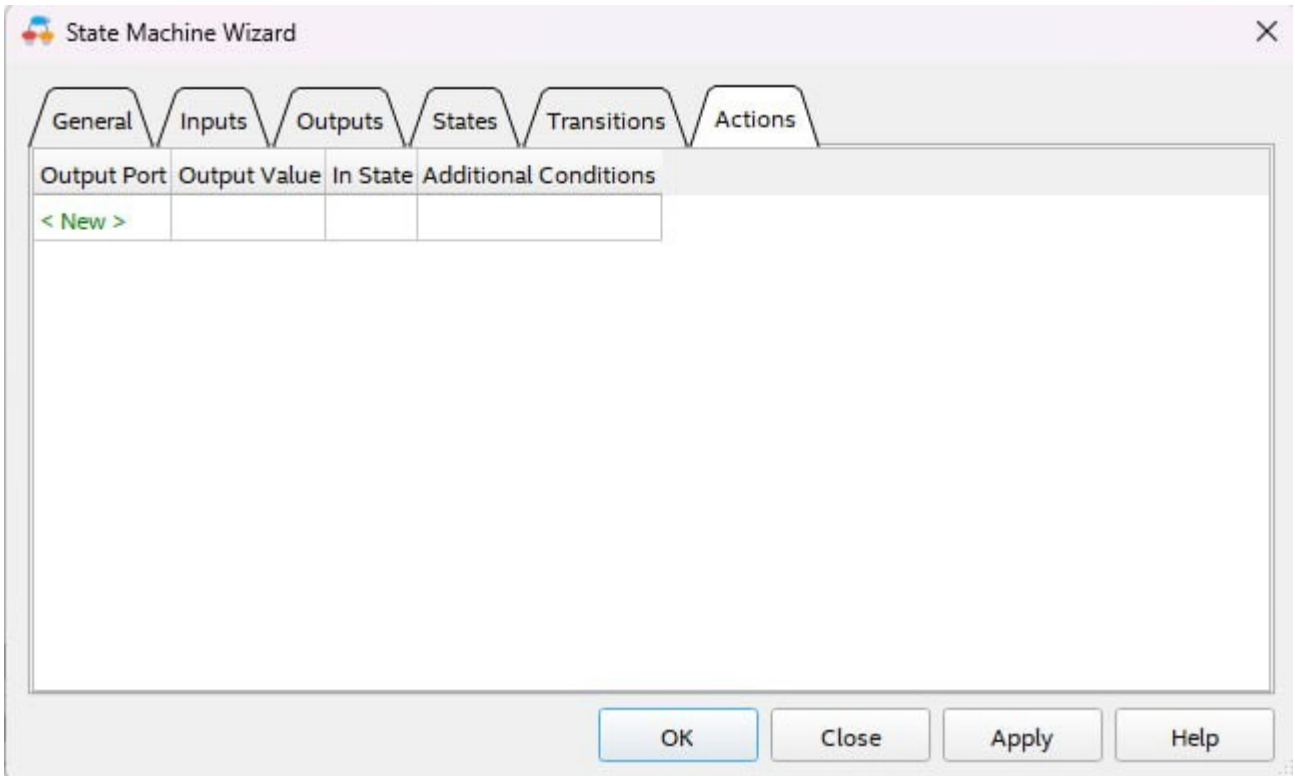
La siguiente pestaña nos pide las transiciones entre nuestros estados.



Para introducir los cambios tenemos que utilizar nomenclatura Verilog.



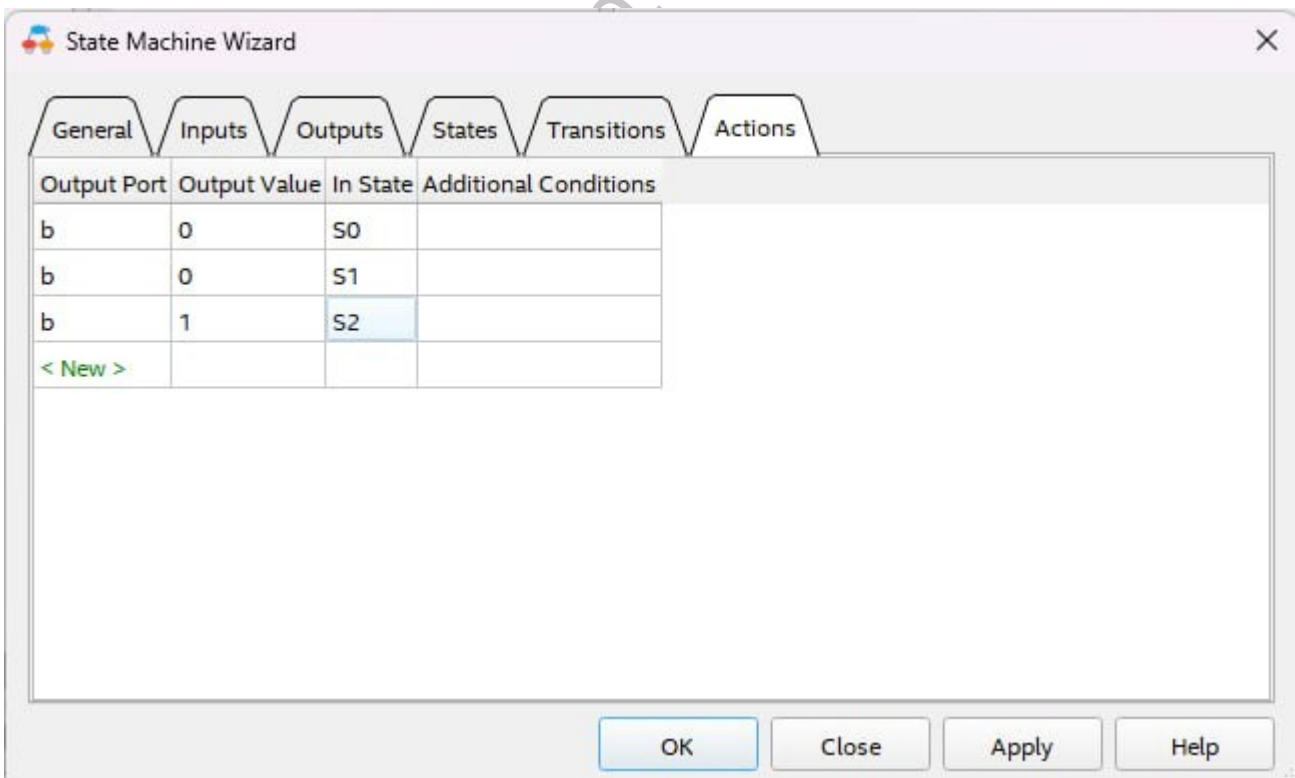
Para terminar nos pide las salidas que se generan en cada estado.



The State Machine Wizard dialog box is shown with the 'General' tab selected. It contains a table with four columns: 'Output Port', 'Output Value', 'In State', and 'Additional Conditions'. The first row of the table is empty and has a green '< New >' button to its left. The dialog box has buttons for 'OK', 'Close', 'Apply', and 'Help' at the bottom right.

Output Port	Output Value	In State	Additional Conditions
< New >			

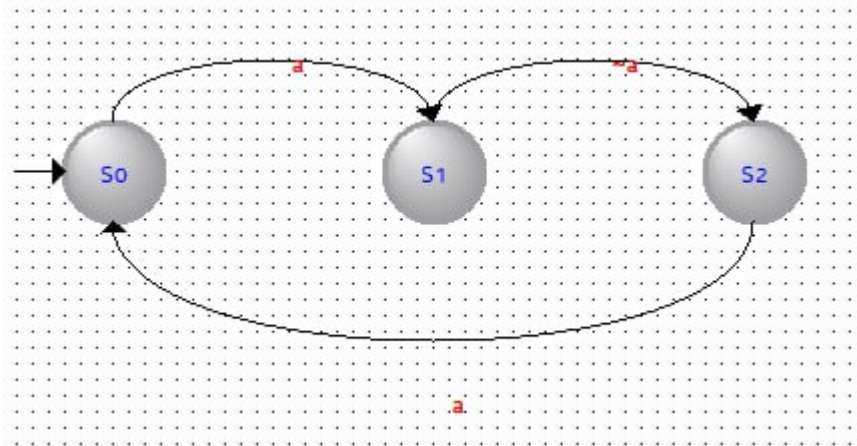
Entonces, de las salidas que hemos definido en salidas le tenemos que decir el valor que toma la salida en cada estado.



The State Machine Wizard dialog box is shown with the 'General' tab selected. It contains a table with four columns: 'Output Port', 'Output Value', 'In State', and 'Additional Conditions'. The first row of the table is empty and has a green '< New >' button to its left. The dialog box has buttons for 'OK', 'Close', 'Apply', and 'Help' at the bottom right.

Output Port	Output Value	In State	Additional Conditions
b	0	S0	
b	0	S1	
b	1	S2	
< New >			

Para terminar le damos a OK y nos aparece el siguiente diagrama con nuestra máquina.



Ahora generamos el código, para ello le damos al botón que pone «HDL».

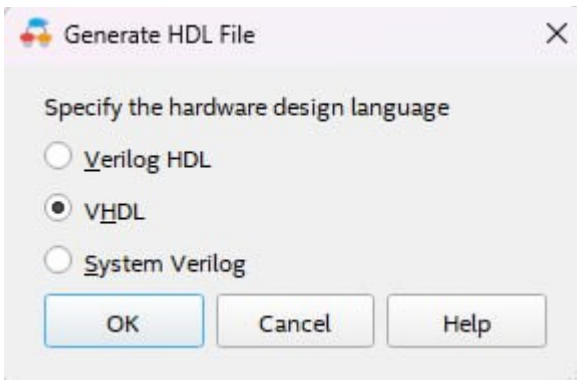


Lo primero que nos pide es guardar el diagrama de la máquina de estados.

Nombre:

Tipo:

Después nos pregunta por el lenguaje en el que generar el código.



Al terminar nos genera el código de la máquina de estados. Es muy posible que este código funcione, pero requiera de cambios para ajustarse a estándares avanzados de programación HDL.

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;

ENTITY SM3 IS
    PORT (
        clock : IN STD_LOGIC;
        reset : IN STD_LOGIC := '0';
        a : IN STD_LOGIC := '0';
        b : OUT STD_LOGIC
    );
END SM3;

ARCHITECTURE BEHAVIOR OF SM3 IS
    TYPE type_fstate IS (S0,S1,S2);
    SIGNAL fstate : type_fstate;
    SIGNAL reg_fstate : type_fstate;
BEGIN
    PROCESS (clock,reg_fstate)
    BEGIN
        IF (clock='1' AND clock'event) THEN
            fstate <= reg_fstate;
        END IF;
    END PROCESS;

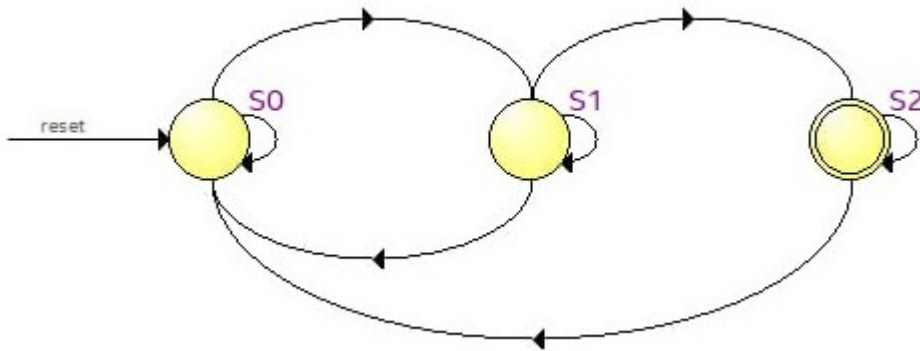
    PROCESS (fstate,reset,a)
    BEGIN
        IF (reset='1') THEN
            reg_fstate <= S0;
            b <= '0';
        ELSE
            b <= '0';
            CASE fstate IS
                WHEN S0 =>
                    IF ((a = '1')) THEN
                        reg_fstate <= S1;
                        -- Inserting 'else' block to prevent latch inference
                    ELSE
                        reg_fstate <= S0;
                    END IF;

                    b <= '0';
                WHEN S1 =>
                    IF (NOT((a = '1')))) THEN
                        reg_fstate <= S2;
                        -- Inserting 'else' block to prevent latch inference
                    ELSE
                        reg_fstate <= S1;
                    END IF;

                    b <= '0';
                WHEN S2 =>
                    IF ((a = '1')) THEN
                        reg_fstate <= S0;
                        -- Inserting 'else' block to prevent latch inference
                    ELSE
                        reg_fstate <= S2;
                    END IF;

                    b <= '1';
                WHEN OTHERS =>
                    b <= 'X';
                    report "Reach undefined state";
            END CASE;
        END IF;
    END PROCESS;
END BEHAVIOR;
```

NOTA: abajo de los diagramas de estados puedes modificar parámetros de las máquinas de estados. Ahora si analizamos la máquina de estados que sintetiza, se puede ver que es correcta.



NOTA: en esta entrada tienes como se utiliza.

<https://soceame.wordpress.com/2025/01/15/como-utilizar-el-analizador-de-maquinas-de-estados-de-quartus/>