VHDL: Máquinas de estado

Microarquitecturas y Softcores

Laboratorio de Sistemas Embebidos

FACULTAD DE INGENIERIA
Universidad de Buenos Aires

Máquinas de estado (FSM)

- **Definición**

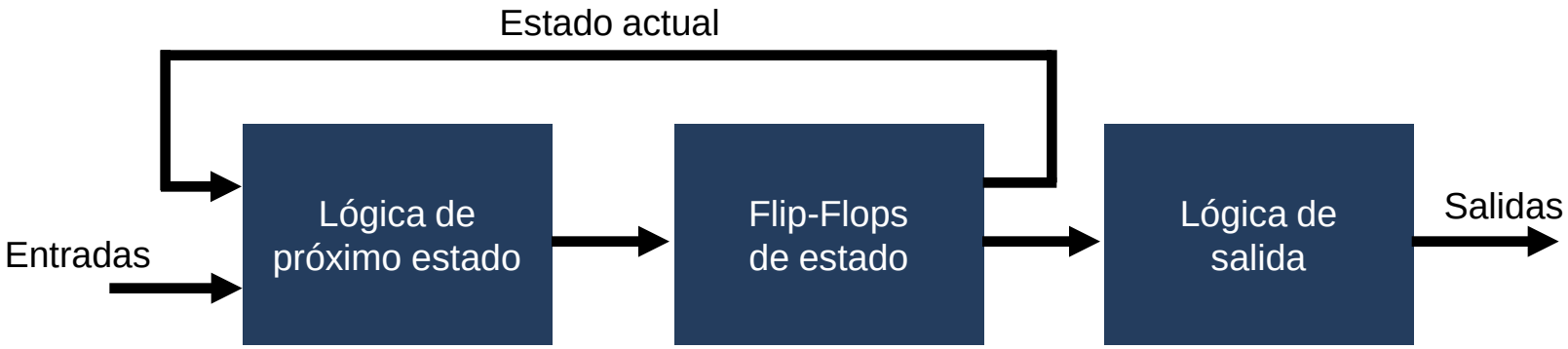
  Una máquina de estados finita (FSM) es un modelo usado para diseñar circuitos lógicos secuenciales.

- **Características**
  - No puede estar en más de un estado por vez.
  - El estado en el que se encuentra se denomina estado actual.
  - El cambio de un estado a otro se denomina transición, y se dispara con el reloj del circuito
  - Son muy útiles en el diseño de protocolos de comunicación.
  - Existen dos tipos de máquinas de estado: Moore y Mealy

Máquinas de estado (FSM)

- **Estructura de una máquina de Moore**

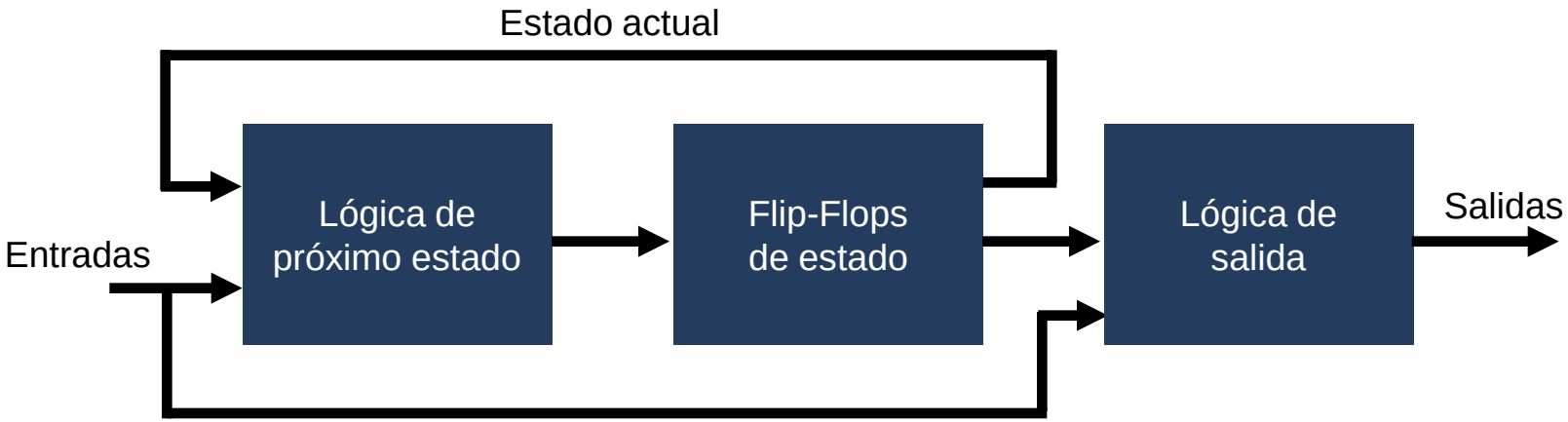
  Las salidas dependen sólo del estado.



Máquinas de estado (FSM)

- **Estructura de una máquina de Mealy**

Las salidas dependen del estado y de las entradas.



Máquinas de estado (FSM)

- **Codificación de estados**
  - **Existen diferentes tipos de codificación de estados**
    - Binaria
    - One Hot
    - Gray
    - etc
  - **La codificación de estados puede hacerse manualmente o de manera automática**
    - En el mismo código de VHDL se pueden establecer atributos que indican que tipo de codificación se quiere
    - Se puede indicar a la herramienta de síntesis como codificar los estados.

Máquinas de estado (FSM)

- **Posibles codificaciones para 8 estados**

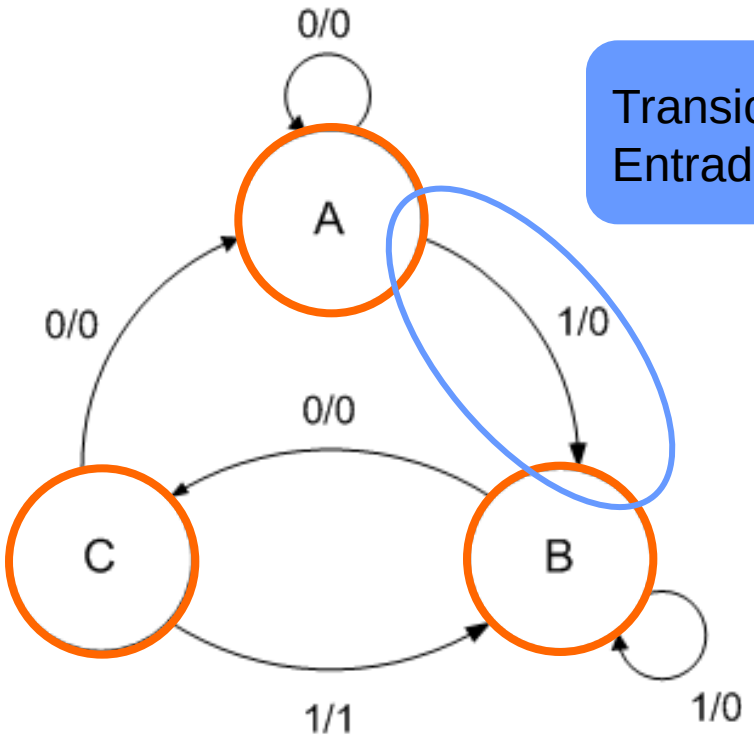
	Encoding		
State	Binary Style	One-Hot Style	Two-Hot Style
State1	000	00000001	00011
State2	001	00000010	00101
State3	010	00000100	01001
State4	011	00001000	10001
State5	100	00010000	00110
State6	101	00100000	01010
State7	110	01000000	10010
State8	111	10000000	01100
Flip-flops number	three flip-flops	eight flip-flops	five flip-flops

Máquinas de estado (FSM)

- **Diseño e implementación de un detector de la secuencia “101” utilizando flip-flops D**

Paso 1: Construir el diagrama de estados

Estados



Transición del estado A al B
Entrada 1 / Salida 0

- **Diseño e implementación de un detector de la secuencia “101” utilizando flip-flops D**

Paso 2: Construir la tabla de estados

Estado actual	Entradas: 0	Entradas: 1
A	A/0	B/0
B	C/0	B/0
C	A/0	B/1

Estado siguiente / Salida

- **Diseño e implementación de un detector de la secuencia “101” utilizando flip-flops D**

Paso 3: Codificar los estados

A = 00

B = 01

C = 10

** Nota: cada estado estará representado por 2 bits Q_1 Q_0*

- **Diseño e implementación de un detector de la secuencia “101” utilizando flip-flops D**

Paso 4: Incluir la codificación en la tabla y calcular las funciones de excitación y de salida

Cálculo de la función de excitación del flip-flop 0

Estado actual	Entradas: 0	Entradas: 1
0 0	0 0 / 0	0 1 / 0
0 1	1 0 / 0	0 1 / 0
1 0	0 0 / 0	0 1 / 1

D_0

E \ Q_1Q_0	00	01	11	10
0	0	0	X	0
1	1	1	X	1

$D_0 = E$

- **Diseño e implementación de un detector de la secuencia “101” utilizando flip-flops D**

Paso 4: Incluir la codificación en la tabla y calcular las funciones de excitación y de salida

Cálculo de la función de excitación del flip-flop 1

Estado actual	Entradas: 0	Entradas: 1
0 0	0 0 / 0	0 1 / 0
0 1	1 0 / 0	0 1 / 0
1 0	0 0 / 0	0 1 / 1

D_1

E \ Q_1Q_0	00	01	11	10
0	0	1	X	0
1	0	0	X	0

D_1 = not(E) and Q_0

- **Diseño e implementación de un detector de la secuencia “101” utilizando flip-flops D**

Paso 4: Incluir la codificación en la tabla y calcular las funciones de excitación y de salida

Cálculo de la función de salida

Estado actual	Entradas: 0	Entradas: 1
0 0	0 0 / 0	0 1 / 0
0 1	1 0 / 0	0 1 / 0
1 0	0 0 / 0	0 1 / 1

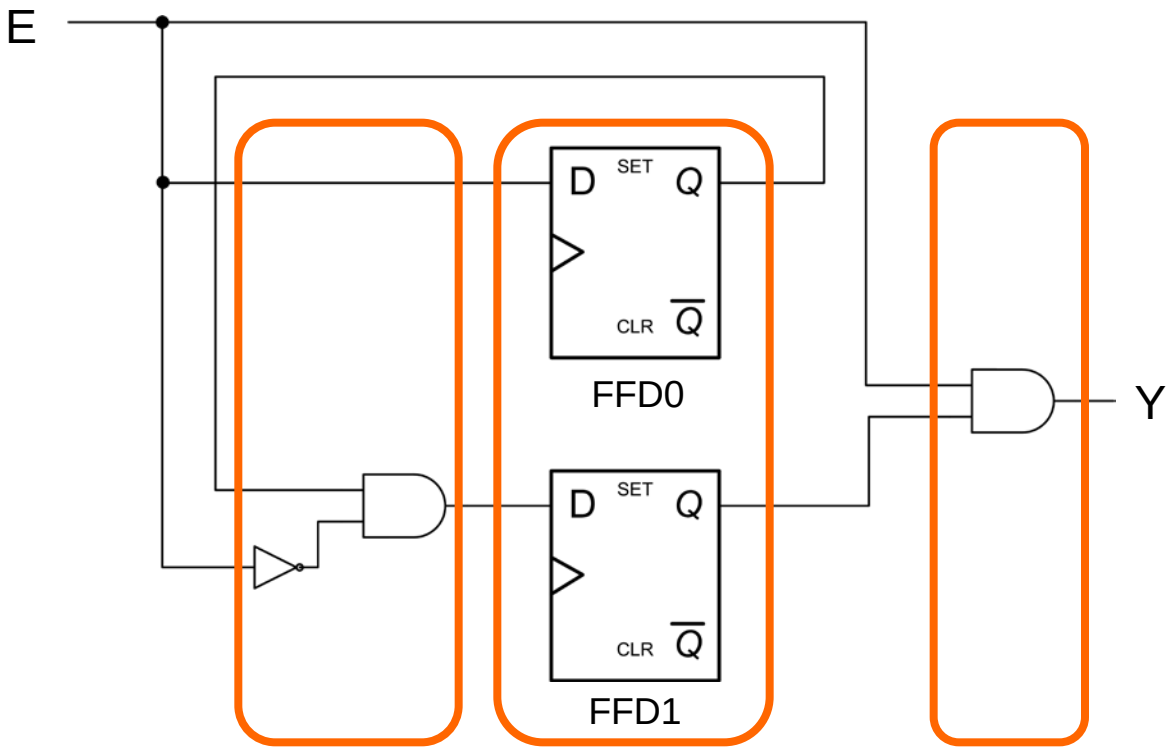
Y

E \ Q_1Q_0	00	01	11	10
0	0	0	X	0
1	0	0	X	1

$Y = E \text{ and } Q_1$

- **Diseño e implementación de un detector de la secuencia “101” utilizando flip-flops D**

Paso 5: Dibujar el circuito



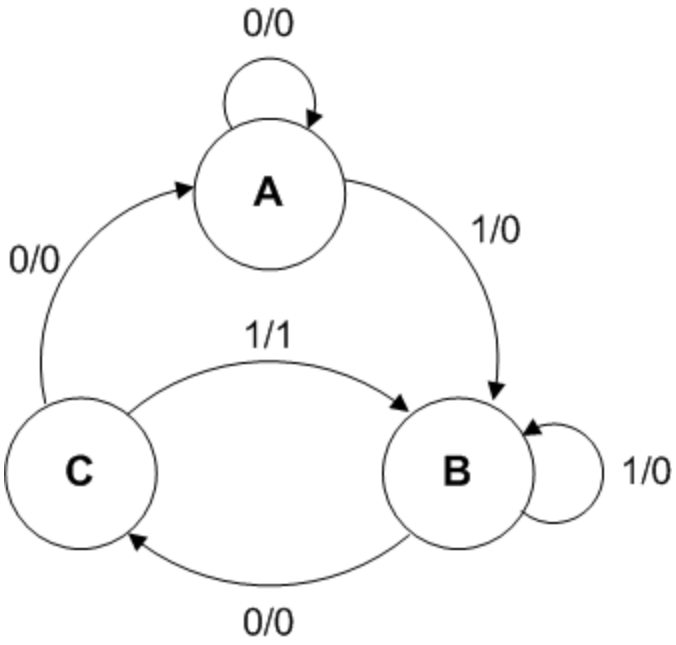
Máquinas de estado (FSM)

Repetimos el ejercicio pero esta vez describimos en VHDL el diagrama de estados en lugar de realizar el diseño a nivel compuertas

Diseñaremos una máquina de Mealy y otra de Moore

Máquinas de estado (FSM)

- **Diseño de una máquina de Mealy: Detector de “101”**



Máquinas de estado (FSM)

- **Diseño de una máquina de Mealy: Detector de “101”**

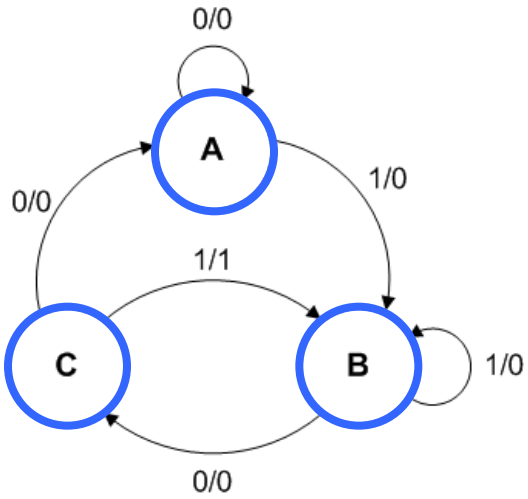
  Consiste en 3 pasos:

  - Definición de los estados
  - Descripción de los registros de estado
  - Descripción de los bloques combinacionales de próximo estado y de salida

- **Diseño de una máquina de Mealy: Detector de “101”**

Paso 1: Definición de los estados

```
architecture det_arq of detector is
   type tipo_estado is (A, B, C);
   signal estado_actual, estado_siguiente: tipo_estado;
begin
   …
end;
```



- **Diseño de una máquina de Mealy: Detector de “101”**

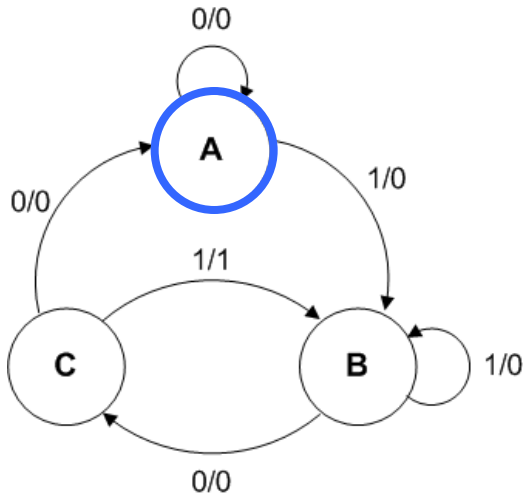
Paso 2: Descripción de los registros de estado

```
registros: process(clk, rst)
    begin
        if rst = '1' then
            estado_actual <= A;
        elsif rising_edge(clk) then
            estado_actual <= estado_siguiente;
        end if;
end process;
```

- **Diseño de una máquina de Mealy: Detector de “101”**

Paso 3: Descripción de los bloques combinacionales de próximo estado y de salida

```
transiciones: process(estado_actual, secuencia)
begin
   case estado_actual is
      when A =>
         if secuencia = '1' then
            estado_siguiente <= B;
            det_flag <= '0';
         else
            estado_siguiente <= A;
            det_flag <= '0';
         end if;
          …
```



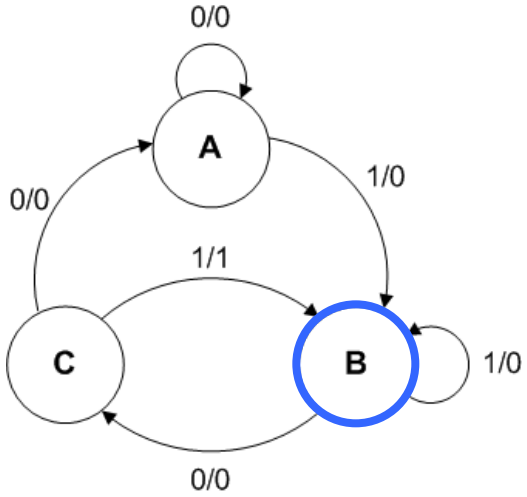
Máquinas de estado (FSM)

- **Diseño de una máquina de Mealy: Detector de “101”**

```
...

when B =>
    if secuencia = '1' then
        estado_siguiente <= B;
        det_flag <= '0';
    else
        estado_siguiente <= C;
        det_flag <= '0';
    end if;

...
```

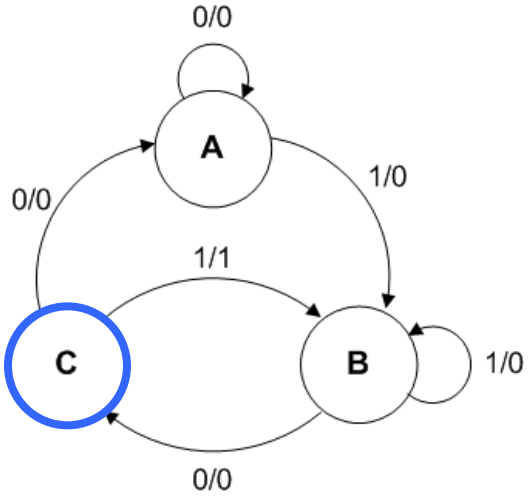


Máquinas de estado (FSM)

- **Diseño de una máquina de Mealy: Detector de “101”**

```
        ...

          when C =>
              if secuencia = '1' then
                  estado_siguiente <= B;
                  det_flag <= '1';
              else
                  estado_siguiente <= A;
                  det_flag <= ‘0';
              end if;
          end case;
      end process;
  end;
```



Máquinas de estado (FSM)

- **Reporte de síntesis (ISE)**

```
=========================================================================
*                            HDL Synthesis                              *
=========================================================================

Performing bidirectional port resolution...

Synthesizing Unit <detector>.
    Related source file is "/home/nalvare/VHDL_sources/Maquinas_de_estado/Ejemplo_basico/prul.vhdl".
    Found finite state machine <FSM_0> for signal <estado_actual>.
    -----------------------------------------------------------------------
    | States             | 3                                              |
    | Transitions        | 6                                              |
    | Inputs             | 1                                              |
    | Outputs            | 3                                              |
    | Clock              | clk                        (rising_edge)       |
    | Reset              | rst                        (positive)          |
    | Reset type         | asynchronous                                   |
    | Reset State        | a                                              |
    | Power Up State     | a                                              |
    | Encoding           | automatic                                      |
    | Implementation     | LUT                                            |
    -----------------------------------------------------------------------
    Summary:
          inferred   1 Finite State Machine(s).
Unit <detector> synthesized.
```

Máquinas de estado (FSM)

- **Reporte de síntesis (ISE)**

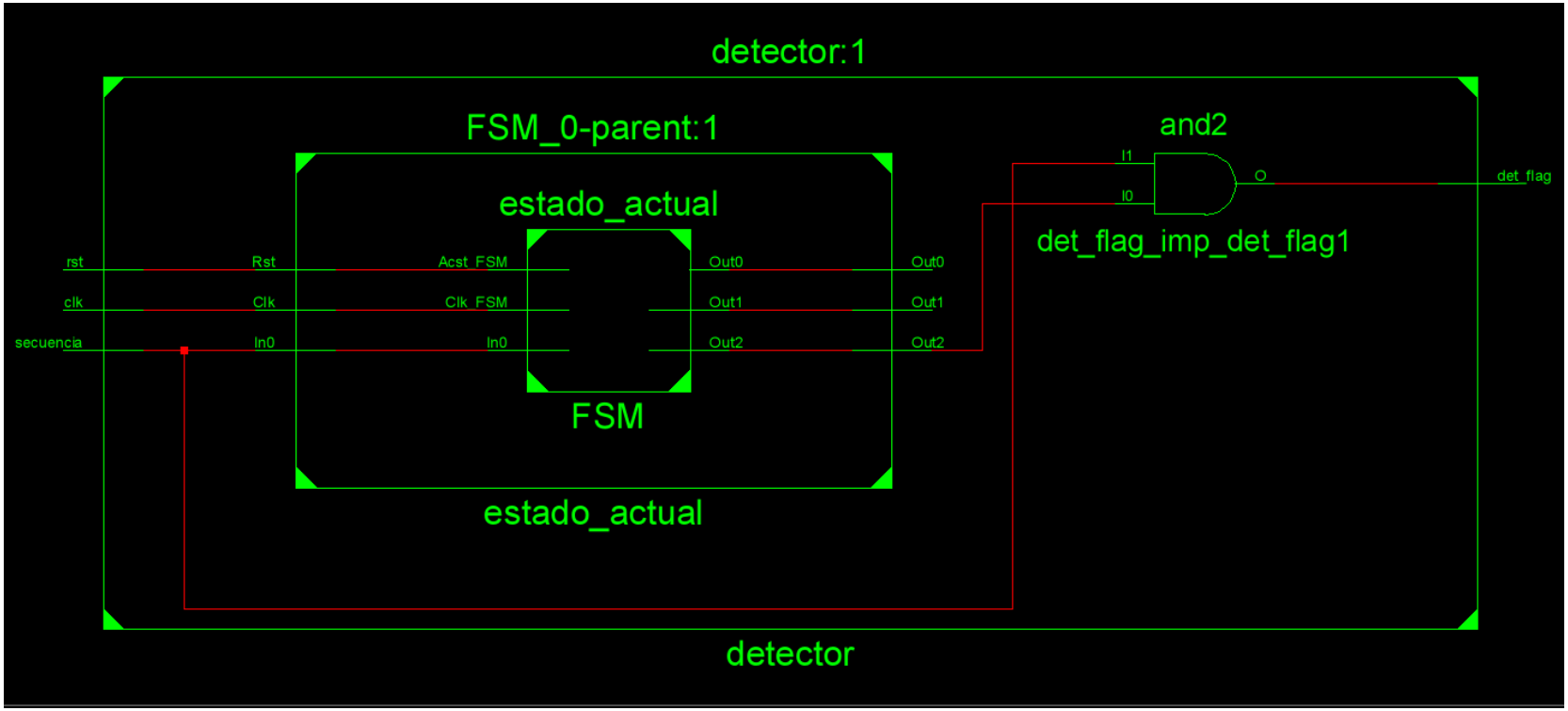
```
=========================================================================
*                        Advanced HDL Synthesis                         *
=========================================================================

Analyzing FSM <FSM_0> for best encoding.
Optimizing FSM <estado_actual/FSM> on signal <estado_actual[1:2]> with user encoding.
-------------------
 State | Encoding
-------------------
 a     | 00
 b     | 01
 c     | 10
-------------------

=========================================================================
```

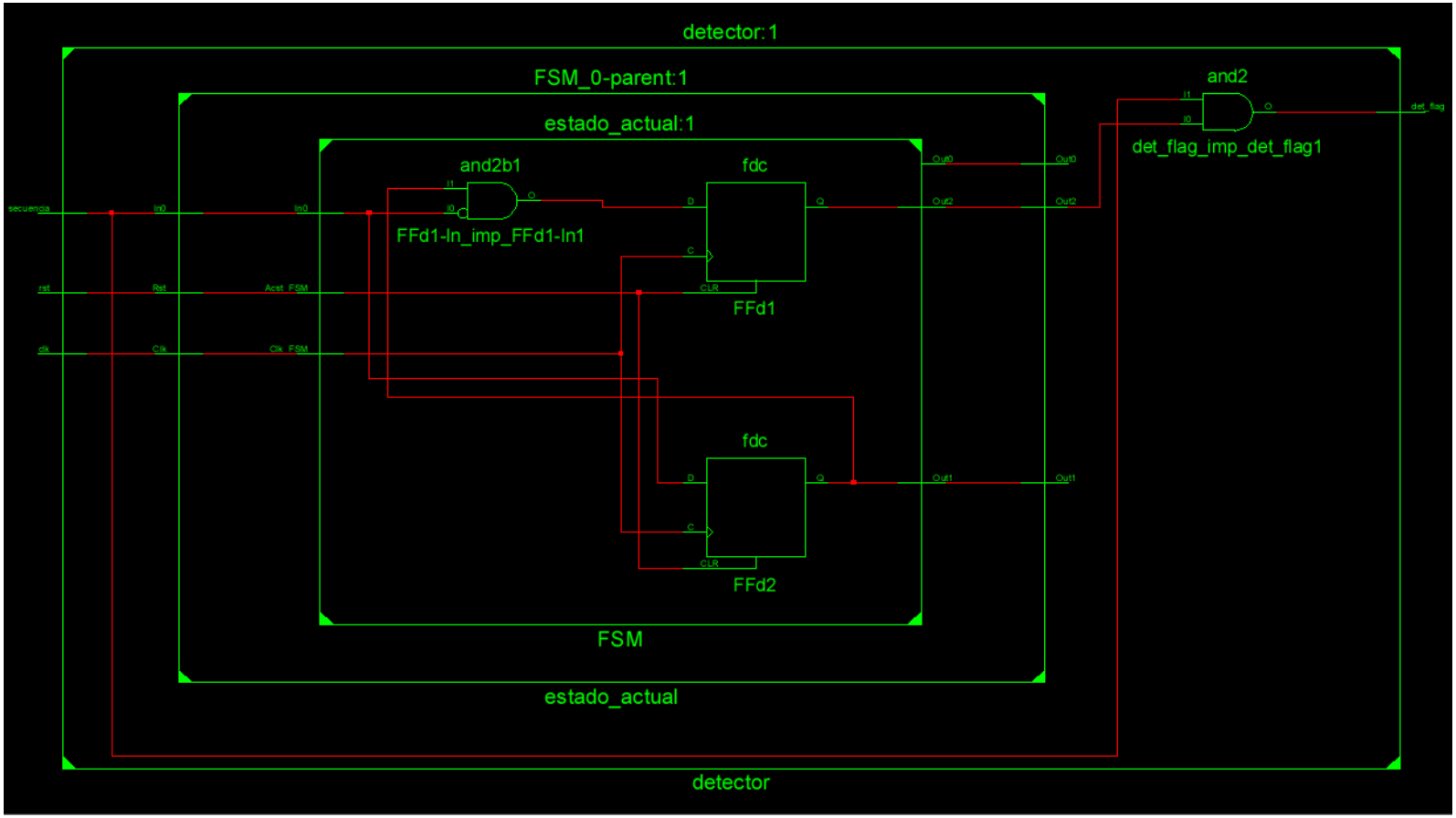
Máquinas de estado (FSM)

- **Esquemático (ISE - RTL Schematic)**



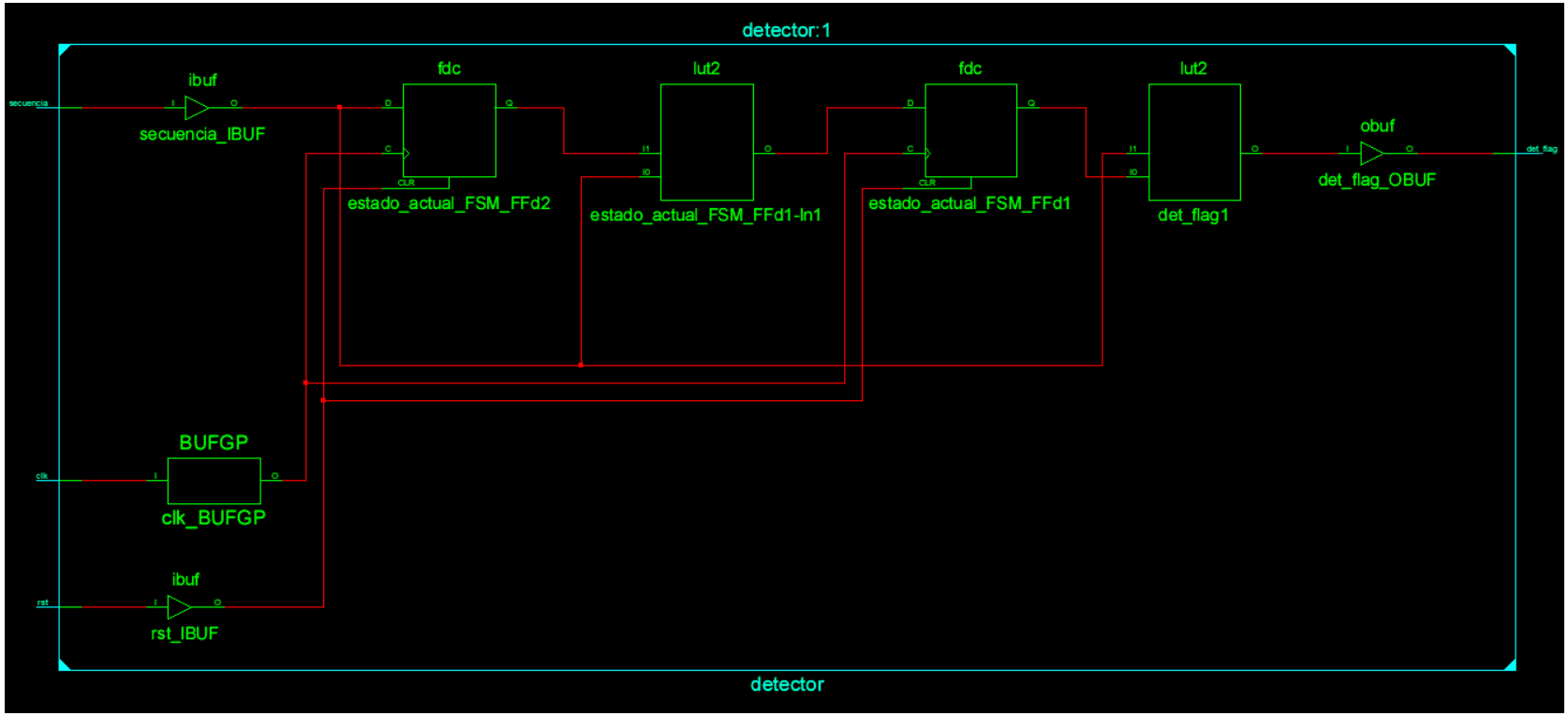
Máquinas de estado (FSM)

- Esquemático (ISE - RTL Schematic)



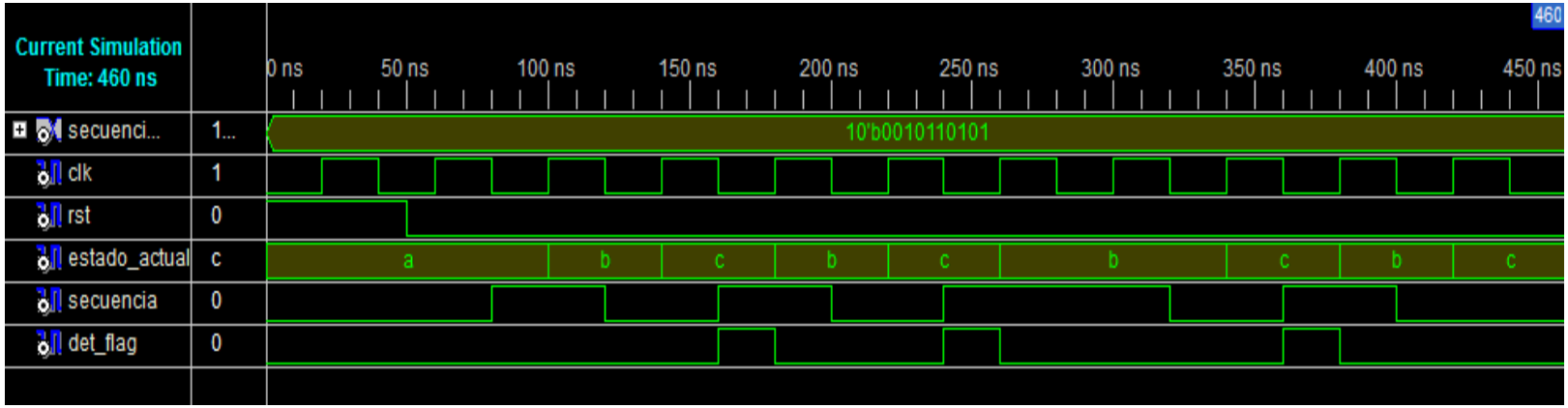
Máquinas de estado (FSM)

- Esquemático (ISE - Technology Schematic)



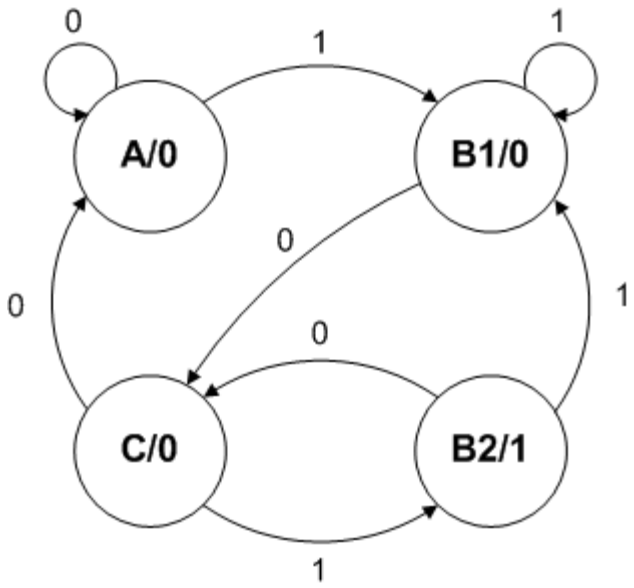
Máquinas de estado (FSM)

- **Diseño de una máquina de Mealy: Simulación (ISim)**



Máquinas de estado (FSM)

- **Diseño de una máquina de Moore: Detector de “101”**



Máquinas de estado (FSM)

- **Diseño de una máquina de Moore: Detector de “101”**

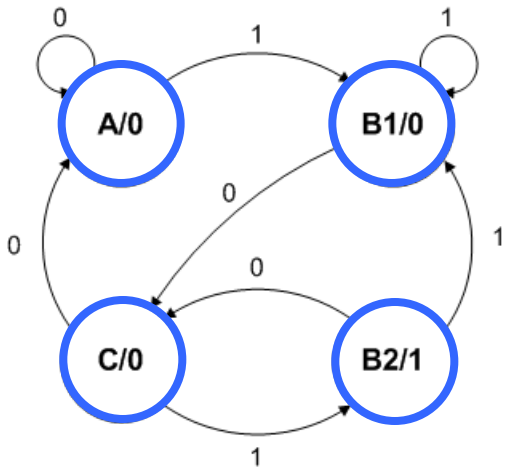
  Consiste en 3 pasos:

  - Definición de los estados
  - Descripción de los registros de estado
  - Descripción de los bloques combinacionales de próximo estado y de salida

- **Diseño de una máquina de Moore: Detector de “101”**

Paso 1: Definición de los estados

```
architecture det_arq of detector is
    type tipo_estado is (A, B1, B2, C);
    signal estado_actual, estado_siguiente: tipo_estado;
begin
   …
end;
```



- **Diseño de una máquina de Moore: Detector de “101”**

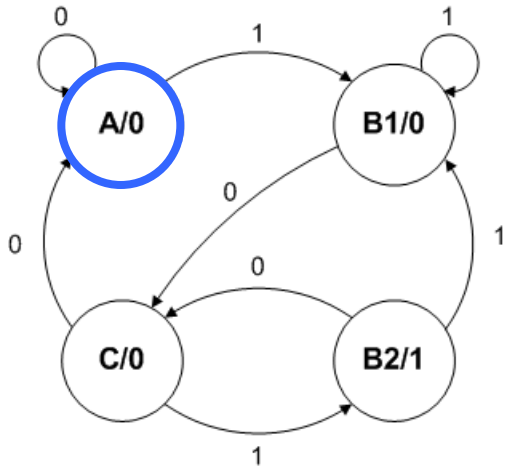
Paso 2: Descripción de los registros de estado

```
registros: process(clk, rst)
    begin
        if rst = '1' then
            estado_actual <= A;
        elsif rising_edge(clk) then
            estado_actual <= estado_siguiente;
        end if;
end process;
```

- **Diseño de una máquina de Moore: Detector de “101”**

Paso 3: Descripción de los bloques combinacionales de próximo estado y de salida

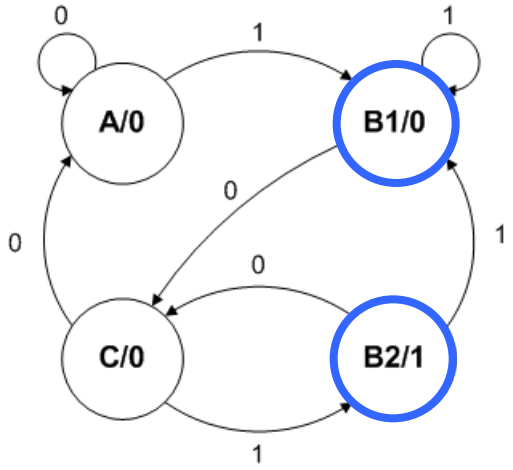
```
transiciones: process(estado_actual, secuencia)
begin
   case estado_actual is
      when A =>
         if secuencia = '1' then
            estado_siguiente <= B1;
         else
            estado_siguiente <= A;
         end if;
      …
```



Máquinas de estado (FSM)

- **Diseño de una máquina de Moore: Detector de “101”**

```
...
when B1 =>
    if secuencia = '1' then
        estado_siguiente <= B1;
    else
        estado_siguiente <= C;
    end if;
when B2 =>
    if secuencia = '1' then
        estado_siguiente <= B1;
    else
        estado_siguiente <= C;
    end if;
...
```



Máquinas de estado (FSM)

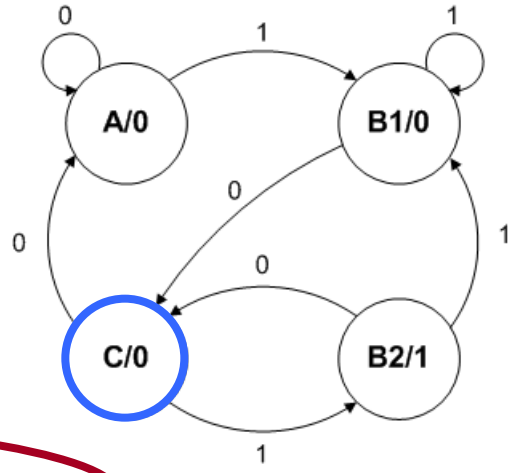
- **Diseño de una máquina de Moore: Detector de “101”**

```
    ...

      when C =>
          if secuencia = '1' then
              estado_siguiente <= B2;
          else
              estado_siguiente <= A;
          end if;
      end case;
  end process;

  det_flag <= '1' when estado_actual = B2 else '0';

end;  -- fin de la arquitectura
```



Máquinas de estado (FSM)

- **Reporte de síntesis de la máquina de Moore (ISE)**

```
=========================================================================
*                            HDL Synthesis                              *
=========================================================================

Performing bidirectional port resolution...

Synthesizing Unit <detector>.
    Related source file is "/home/nalvare/VHDL_sources/Maquinas_de_estado/Ejemplo_Basico_Moore/pru_moore.vhd".
    Found finite state machine <FSM_0> for signal <estado_actual>.
    -----------------------------------------------------------------------
    | States             | 4                                              |
    | Transitions        | 8                                              |
    | Inputs             | 1                                              |
    | Outputs            | 1                                              |
    | Clock              | clk                        (rising_edge)       |
    | Reset              | rst                        (positive)          |
    | Reset type         | asynchronous                                   |
    | Reset State        | a                                              |
    | Power Up State     | a                                              |
    | Encoding           | automatic                                      |
    | Implementation     | LUT                                            |
    -----------------------------------------------------------------------
    Summary:
          inferred   1 Finite State Machine(s).
Unit <detector> synthesized.
```

Máquinas de estado (FSM)

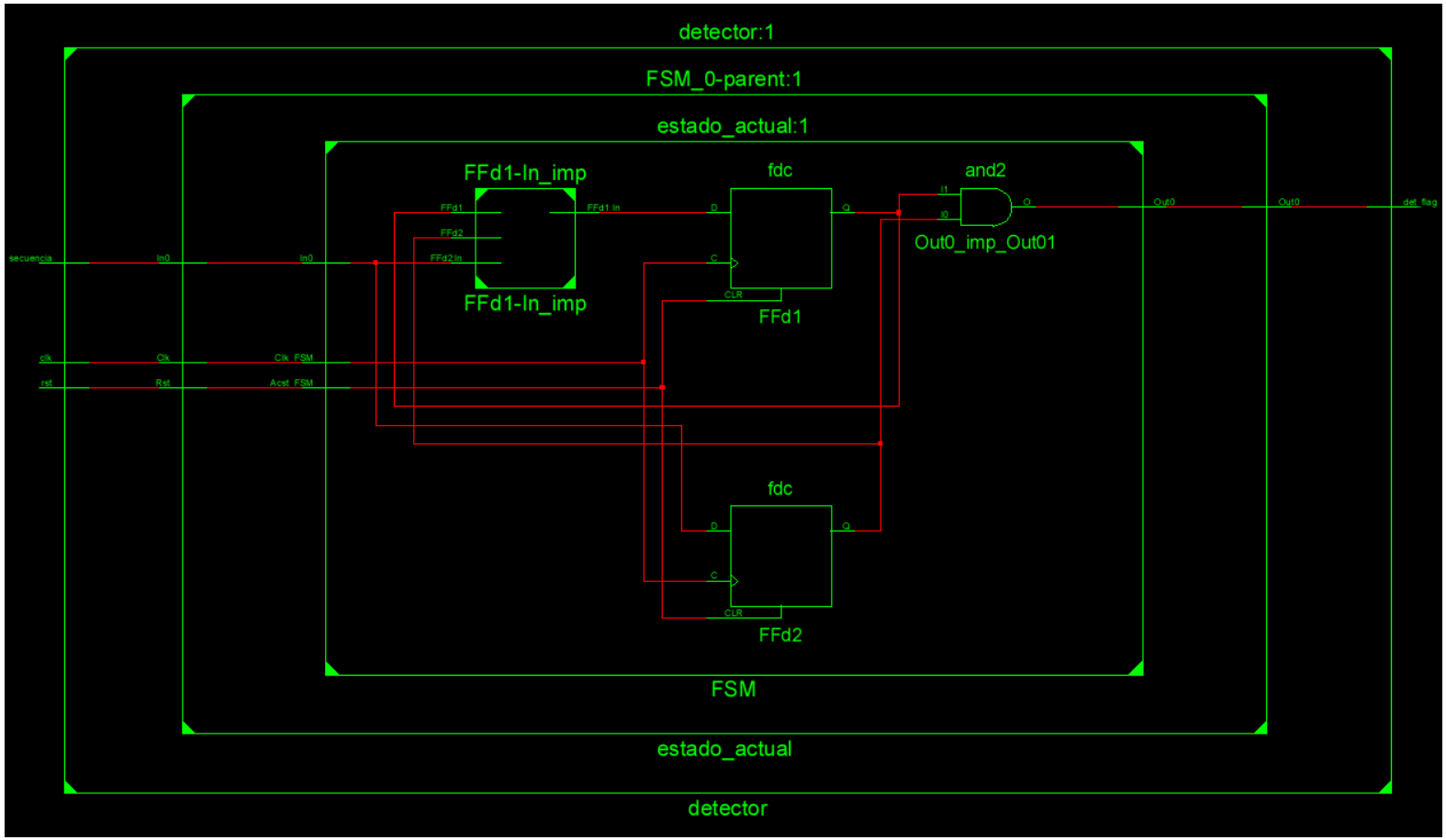
- **Reporte de síntesis de la máquina de Moore (ISE)**

```
=========================================================================
*                        Advanced HDL Synthesis                         *
=========================================================================

Analyzing FSM <FSM_0> for best encoding.
Optimizing FSM <estado_actual/FSM> on signal <estado_actual[1:2]> with sequential encoding.
-------------------
 State | Encoding
-------------------
 a     | 00
 b1    | 01
 b2    | 11
 c     | 10
-------------------
```

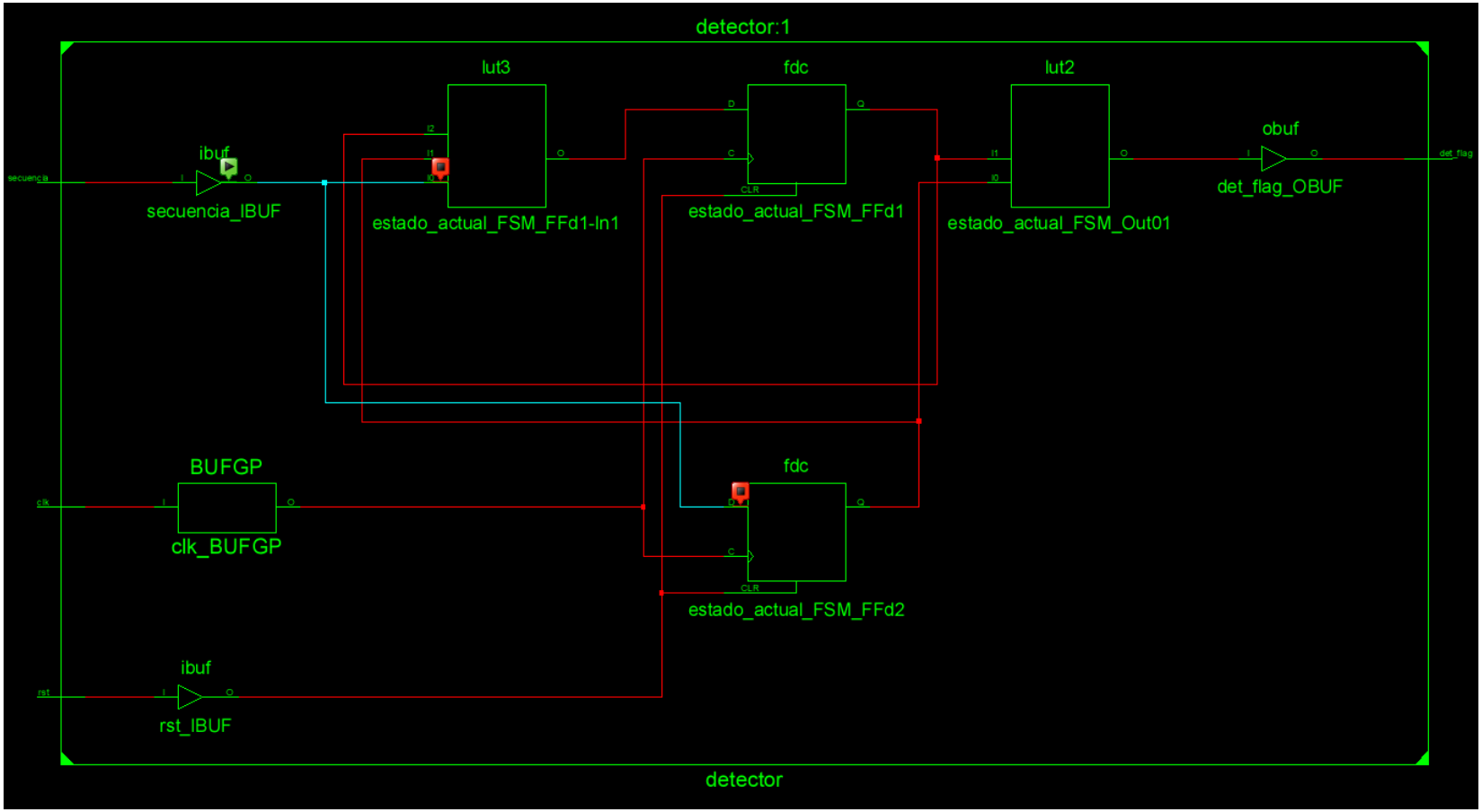
Máquinas de estado (FSM)

- **Esquemático de la máquina de Moore (ISE - RTL Schem)**



Máquinas de estado (FSM)

- Esquemático de la máquina de Moore (ISE - Tech Schem)



Máquinas de estado (FSM)

- **Diseño de una máquina de Moore: Simulación (ISim)**