# Informe Pràctica 3 Sessió 3: Controlador de Bit

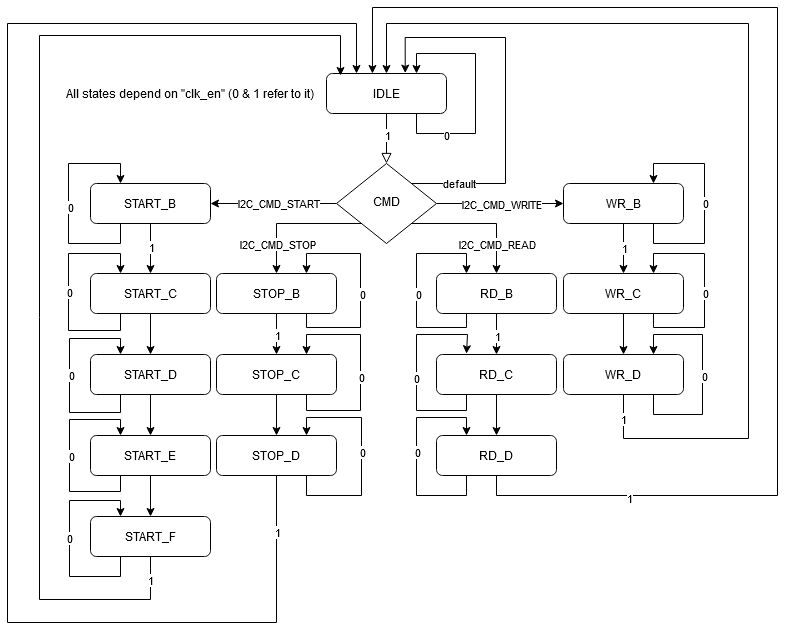
## Errors en el RTL:

|  |  |  |
| --- | --- | --- |
| #Error | Línies | Descripció |
| 1 | 104 | **Descripció de l’error:** Encara que per tal de generar tant una condició de “start” com de “stop” és necessari que SCL estigui en 1, tal com està definida la condició de “start” al codi original no és correcte. Hem canviat ~sSCL per sSCL  **Codi original:**  sta\_condition <= ~sSDA & dSDA & ~sSCL  **Codi corregit:**  sta\_condition <= ~sSDA & dSDA & sSCL |
| 2 | 193  i  198 | **Descripció de l’error:** Al càlcul del següent estat en mode d’escriptura, hi ha un bucle infinit d’estats degut a que se salta l’estat “C” i passa directament al “D”. Després, un cop està a l’estat “D”, la condició *else*, enlloc de mantenir l’estat actual fa un salt a l’estat “B”; generant així un bucle infinit.  **Codi original:**  WR\_B : if(clk\_en) next = WR\_D;  else next = WR\_B;  WR\_C : if(clk\_en) next = WR\_D;  Else next = WR\_C;  WR\_D : if(clk\_en) next = IDLE;  else next = WR\_B;  **Codi corregit:**  WR\_B : if(clk\_en) next = WR\_C;  else next = WR\_B;  WR\_C : if(clk\_en) next = WR\_D;  Else next = WR\_C;  WR\_D : if(clk\_en) next = IDLE;  else next = WR\_D; |
| 3 | 230 | **Descripció de l’error:** A la seqüència de “start”, a la primera part (B), hi ha un error a l’hora d’assignar el valor de “Sda\_oen” ja que, segons la seqüència del gràfic del guió, a la part B el valor d’aquesta ha de commutar a 1 i en aquest cas està assignat a 0.  **Codi original:**  START\_B : begin  Scl\_oen <= 1'b1; *// keep SCL high*  Sda\_oen <= 1'b0; *// keep SDA high*  sda\_chk <= 1'b0; *// don't check SDA output*  end  **Codi corregit:**  START\_B : begin  Scl\_oen <= 1'b1; *// keep SCL high*  Sda\_oen <= 1'b1; *// keep SDA high*  sda\_chk <= 1'b0; *// don't check SDA output*  end |
| 4 | 239 | **Descripció de l’error:** Aquest és només un error de comentari, ja que el que s’està fent no es correspon amb el comentari associat.  **Codi original:**  START\_D : begin      Scl\_oen <= 1'b1; *// set SCL low*      Sda\_oen <= 1'b0; *// set SDA low*      sda\_chk <= 1'b0; *// don't check SDA output*  end  **Codi corregit:**  START\_D : begin      Scl\_oen <= 1'b1; *// keep SCL high*      Sda\_oen <= 1'b0; *// set SDA low*      sda\_chk <= 1'b0; *// don't check SDA output*  end |
| 5 | 244 | **Descripció de l’error:** Aquest altre és també només un error de comentari, ja que el que s’està fent no es correspon amb el comentari associat.  **Codi original:**  START\_E : begin      Scl\_oen <= 1'b1; *// keep SCL low*      Sda\_oen <= 1'b0; *// keep SDA low*      sda\_chk <= 1'b0; *// don't check SDA output*  end  **Codi corregit:**  START\_E : begin      Scl\_oen <= 1'b1; *// keep SCL high*      Sda\_oen <= 1'b0; *// keep SDA low*      sda\_chk <= 1'b0; *// don't check SDA output*  end |

## Màquina d’estats

A les figues 1 i 2 es mostra el diagrama d’estats de la unitat de control del mestre I2C implementat (un diagrama de flux i el diagrama generat automàticament pel Quartus) i la Taula 1 en hi ha llistats els diferents estats amb una breu descripció. Aquesta unitat de control s’encarrega de gestionar la transmissió dels bits d’informació i de control així com la generació del senyal de rellotge.

De tota manera, però, aquesta unitat de control està dividida en dues parts: el controlador a nivell de byte i el controlador a nivell de bit. Aquesta màquina d’estats s’encarrega de gestionar les comandes a nivell de bit. Aquest controlador de bit s’encarrega d’agafar les comandes de byte proveïdes per l’altre controlador i transmetre-les bit a bit als mòduls que ha de controlar, ja que el canal de comunicació és serial; no en paral·lel.



**Figura 1:** Diagrama d’estats del controlador de Bit de la unitat de control I2C.

Diagrama

Descripción generada automáticamente

**Figura 2:** Diagrama d’estats (generat automàticament pel Quartus) del controlador de Bit de la unitat de control I2C.

**Taula 1:** Llista dels diferents estats de la màquina d’estats implementada. Aquí fem referència a l’habilitació d’outputs però li diem SCL i SDA directament per conveniència.

|  |  |  |
| --- | --- | --- |
| Estat | Descripció | Dibuix |
| IDLE | Aquest és l’estat de repòs. En aquest estat no es fa res més que esperar a que arribi una comanda que dicti quina operació s’ha de començar. |  |
| START\_B | En aquest estat es comença la operació de Start, mantenint tant SCL com SDA en estat alt. En cas de repetició de Start, es força la pujada de SCL. |  |
| START\_C | Aquí es mantenen tant SCL com SDA en estat alt, assegurant així que s’està en un estat desitjat per a començar la seqüència “important” que determinarà un Start. |
| START\_D | Aquí comença la seqüència “important” de Start, canviant SDA a un nivell baix (mantenint SCL) |
| START\_E | En aquest estat es mantenen els valors de l’estat anterior. |
| START\_F | Aquest és l’últim estat per a la seqüència de Start, canviant SCL a baix (mantenint SDA). |
| STOP\_B | En aquest estat es comença la seqüència de Stop, la qual portarà els valors de SCL i SDA als de l’estat d’IDLE. En aquest pas es canvia SCL de baix a alt. |  |
| STOP\_C | Aquí es mantenen els valors de l’estat anterior. |
| STOP\_D | Aquí s’acaba la seqüència de Stop, tornant a canviar SDA a nivell alt i mantenint SCL en nivell baix. |
| RD\_B | En aquest estat s’habilita SCL deixant SDA en tri-estat, permetent que l’emissor enviï el bit d’informació a llegir. |  |
| RD\_C | Aquí es mantenen els valors de l’estat anterior. |
| RD\_D | Aquí es deshabilita SCL finalitzant la seqüència de Read. |
| WR\_B | En aquest estat s’habilita SCL mentre a SDA se li assigna el valor que es desitgi transmetre. |  |
| WR\_C | Aquí es mantenen els valors de l’estat anterior. |
| WR\_D | En aquest estat es finalitza la seqüència de Write, deshabilitant SCL. |

## Qüestions

1. Perquè la generació del senyal Start és de 6 etapes en lloc de 4?

Per tal de gestionar el fet que es pot tornar a començar una seqüència de Start sense haver passat per un Stop abans, cal afegir 2 estats més que permetin el restabliment de la seqüència com si s’estigués fent un Start per primer cop.

1. Quina és la formula que ens dona la freqüència de rellotge o temps de bit?

La freqüència de rellotge (temps de bit) vindrà donat pel *prescaler*, el qual canviarà la freqüència efectiva del sistema. Aquesta freqüència es pot calcular a través de la següent expressió:

SCK = CLK / (2 \* PRER) on PRER es el valor de pre-escalat.

1. Com s’implementa la detecció de coŀlisions? I el clock stretching? Dibuixeu l’esquema RTL de la lògica i mostreu una captura de simulació a nivell RTL que ho mostri com funcionen.