Laboratorio 9:

Diseño con SDRAM off-chip

FSM temporizadas

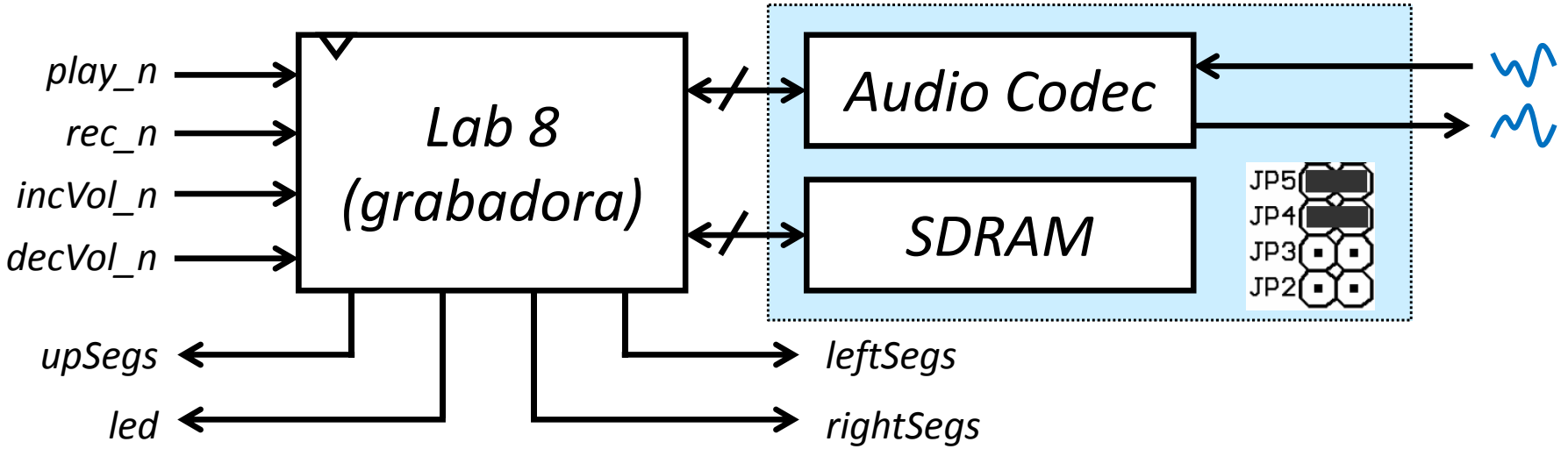
Diseño automático de sistemas

José Manuel Mendías Cuadros
Dpto. Arquitectura de Computadores y Automática
Universidad Complutense de Madrid

J.M. Mendías
2017

Presentación

- Diseñar un grabador/reproductor de sonido muestreado.
  - Para grabar/reproducir usará el audio CODEC de la placa XST.
    - Indicará que está grabando encendiendo un led
  - Las muestras de sonido se almacenarán en la SDRAM de la placa XSA-3S.
  - El comienzo/fin de la grabación/reproducción
    - Se indicará a pulsaciones de los pulsadores 2/3 de la placa XST
  - El tiempo (0-59 segundos) de grabación o restante de reproducción
    - Se visualizará en los el displays 7-seg de la placa XST
  - La reproducción se podrá realizar a volumen variable
    - El nivel de volumen (0-7) se visualizará en el display 7-seg de la placa XSA-3S.
    - Se aumentará/disminuirá a pulsaciones de los pulsadores izq/der de la placa XSA-3S.

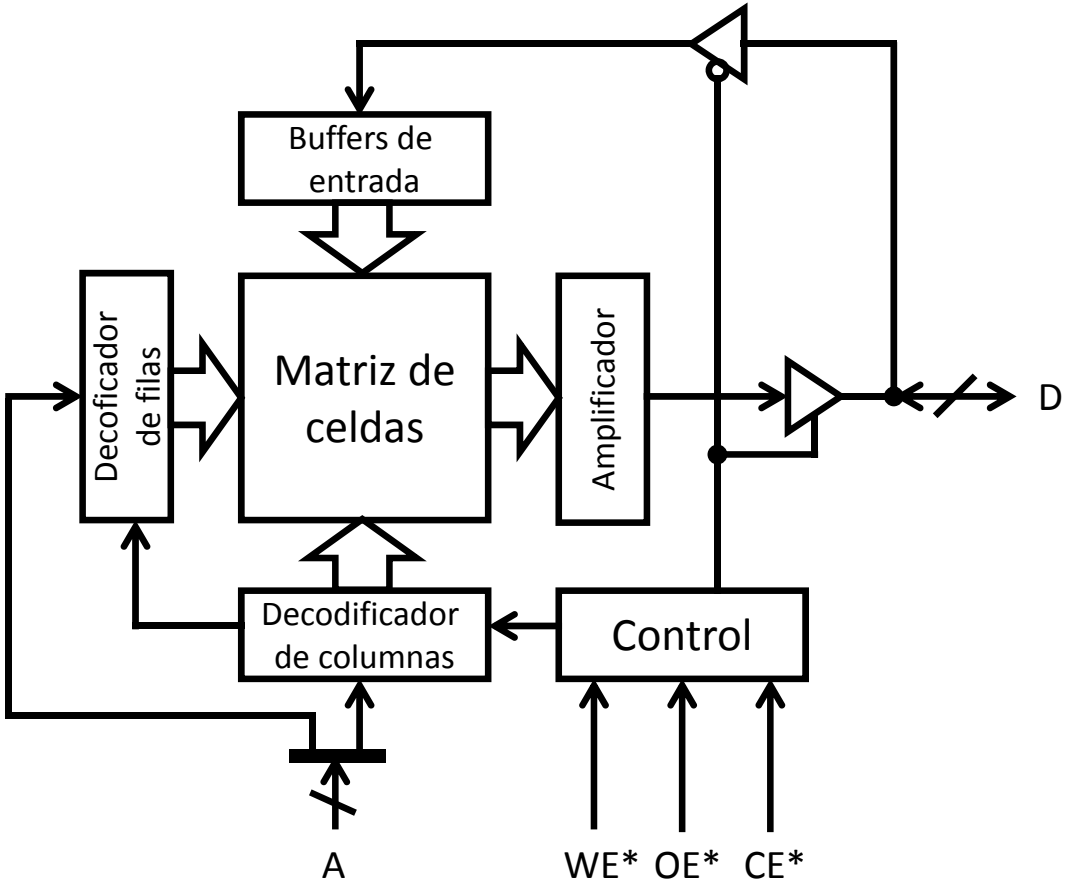


J.M. Mendías
2017

Memorias SRAM

estructura

- Las SRAM son memorias asíncronas de tipo estático
  - La celda elemental es un latch, por lo que no requieren refresco

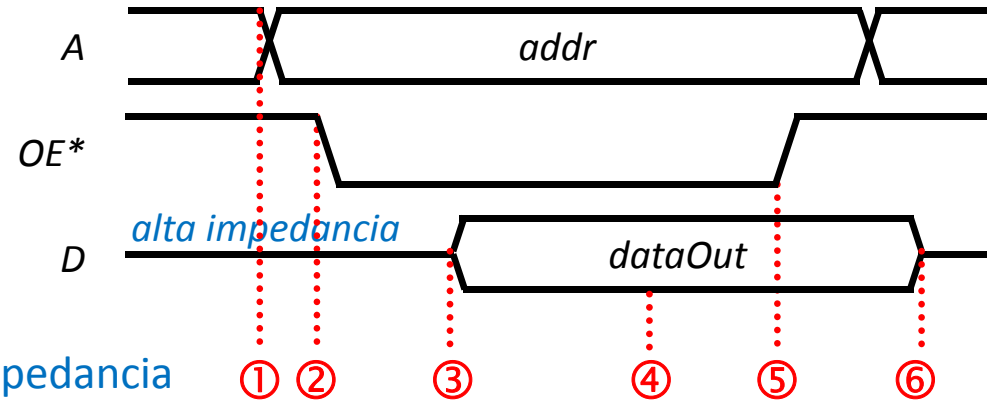


Memorias SRAM

ciclos de acceso

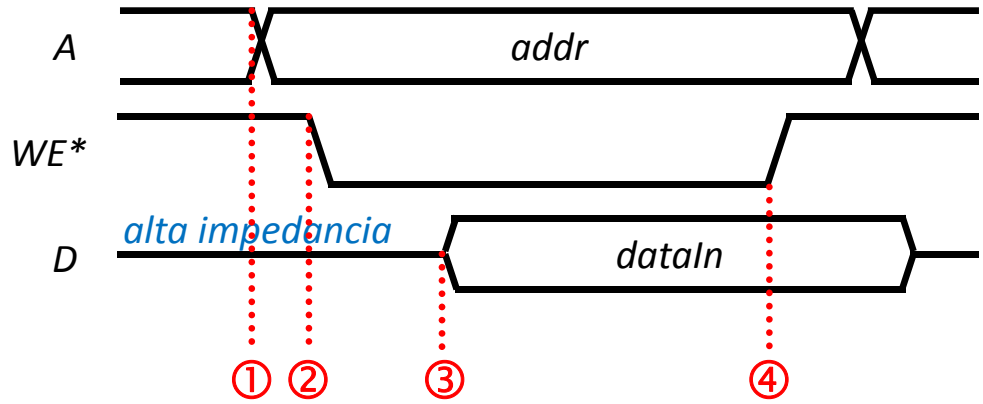
- Ciclo de lectura (dirigido por OE)

1. Escribir dirección
2. Activar OE (a baja)
3. SRAM vuelca dato
4. Leer dato
5. Desactivar OE
6. SRAM pone salida en alta impedancia



- Ciclo de escritura (dirigido por WE):

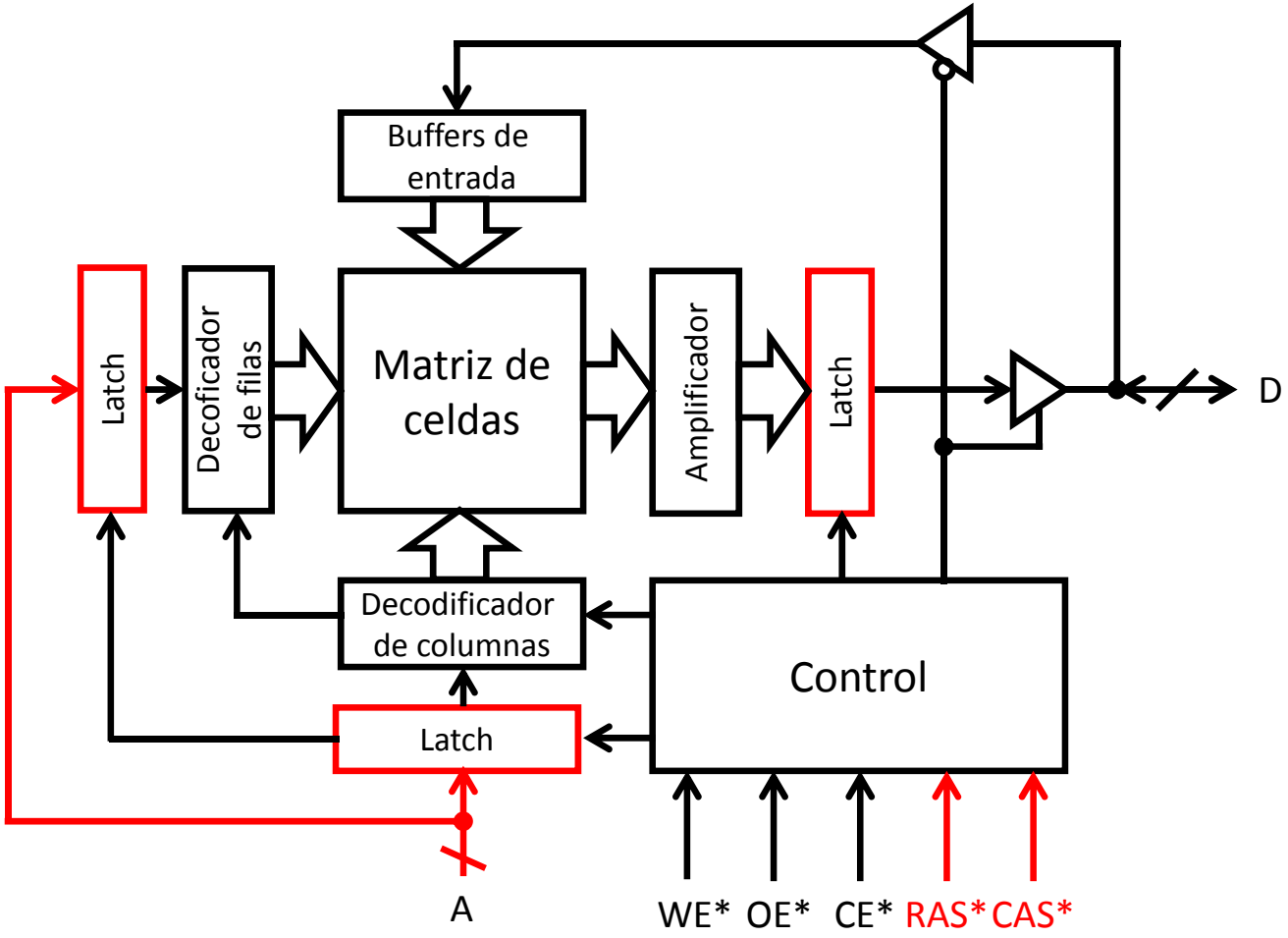
1. Escribir dirección
2. Activar WE (a baja)
3. Escribir dato
4. Desactivar WE



Memorias DRAM

ciclos de acceso

- Las DRAM son memorias asíncronas de tipo dinámico
  - La celda elemental es un transitor+condensador, por lo que requieren refresco.
  - La dirección de fila y columna se multiplexan en el tiempo.
  - Suelen disponer de un buffer de fila que permite la lectura de datos en ráfaga.



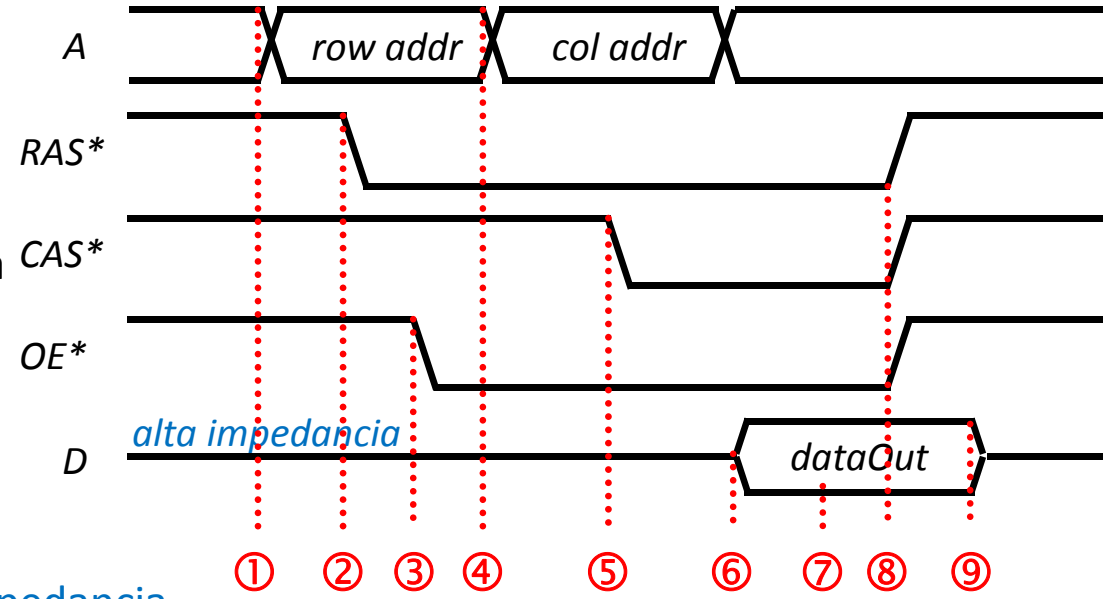
J.M. Mendías
2017

Memorias DRAM

ciclos de acceso (i)

- Ciclo de lectura (early read)

1. Escribir dirección de fila
2. Activar RAS (a baja)
3. Activar OE (a baja)
4. Escribir dirección de columna
5. Activar CAS (a baja)
6. DRAM vuelca dato
7. Leer dato
8. Desactivar OE/RAS/CAS
9. DRAM pone salida en alta impedancia



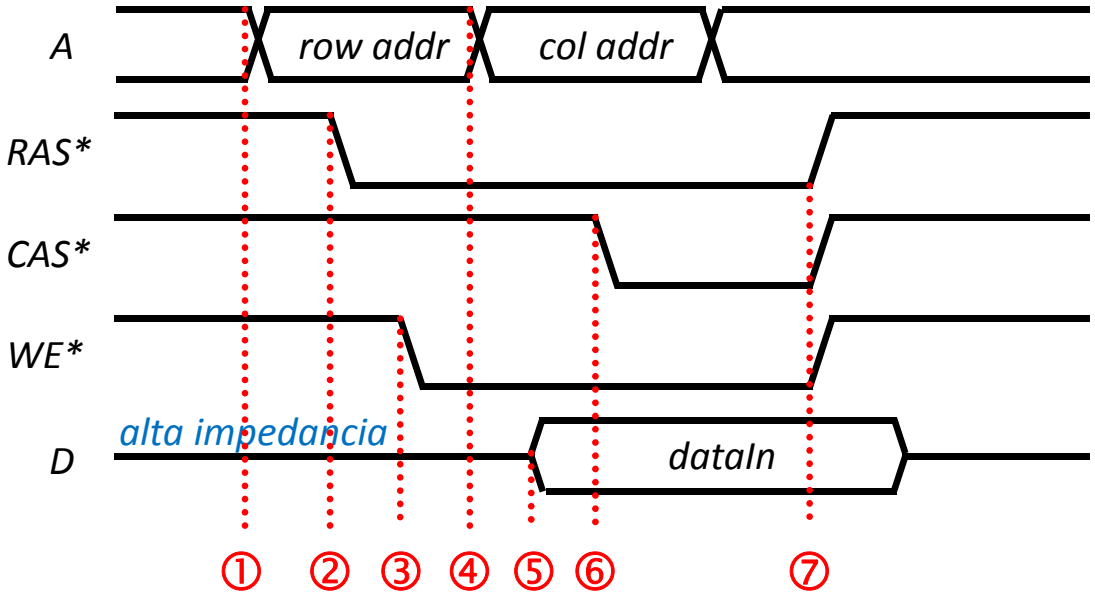
J.M. Mendías 2017

Memorias DRAM

ciclos de acceso (ii)

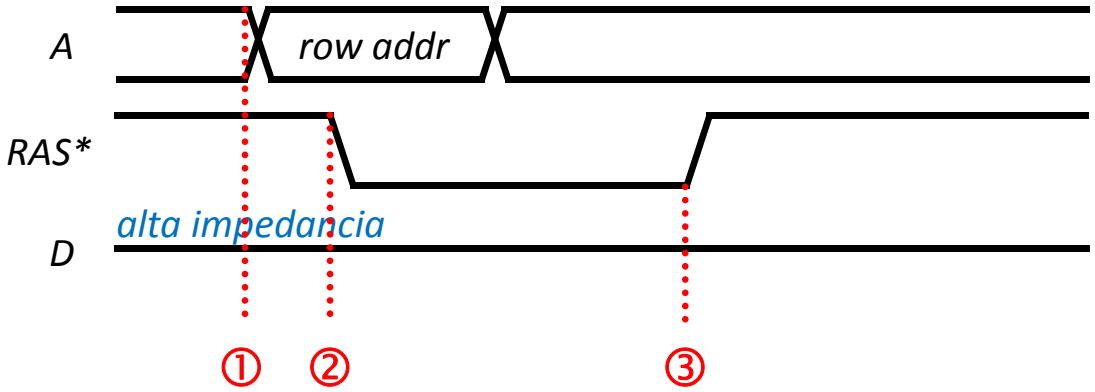
- Ciclo de escritura (early write)

1. Escribir dirección de fila
2. Activar RAS (a baja)
3. Activar WE (a baja)
4. Escribir dirección de columna
5. Escribir dato
6. Activar CAS (a baja)
7. Desactivar OE/RAS/CAS



- Ciclo de refresco (only-RAS)

1. Escribir dirección de fila
2. Activar RAS (a baja)
3. Desactivar RAS



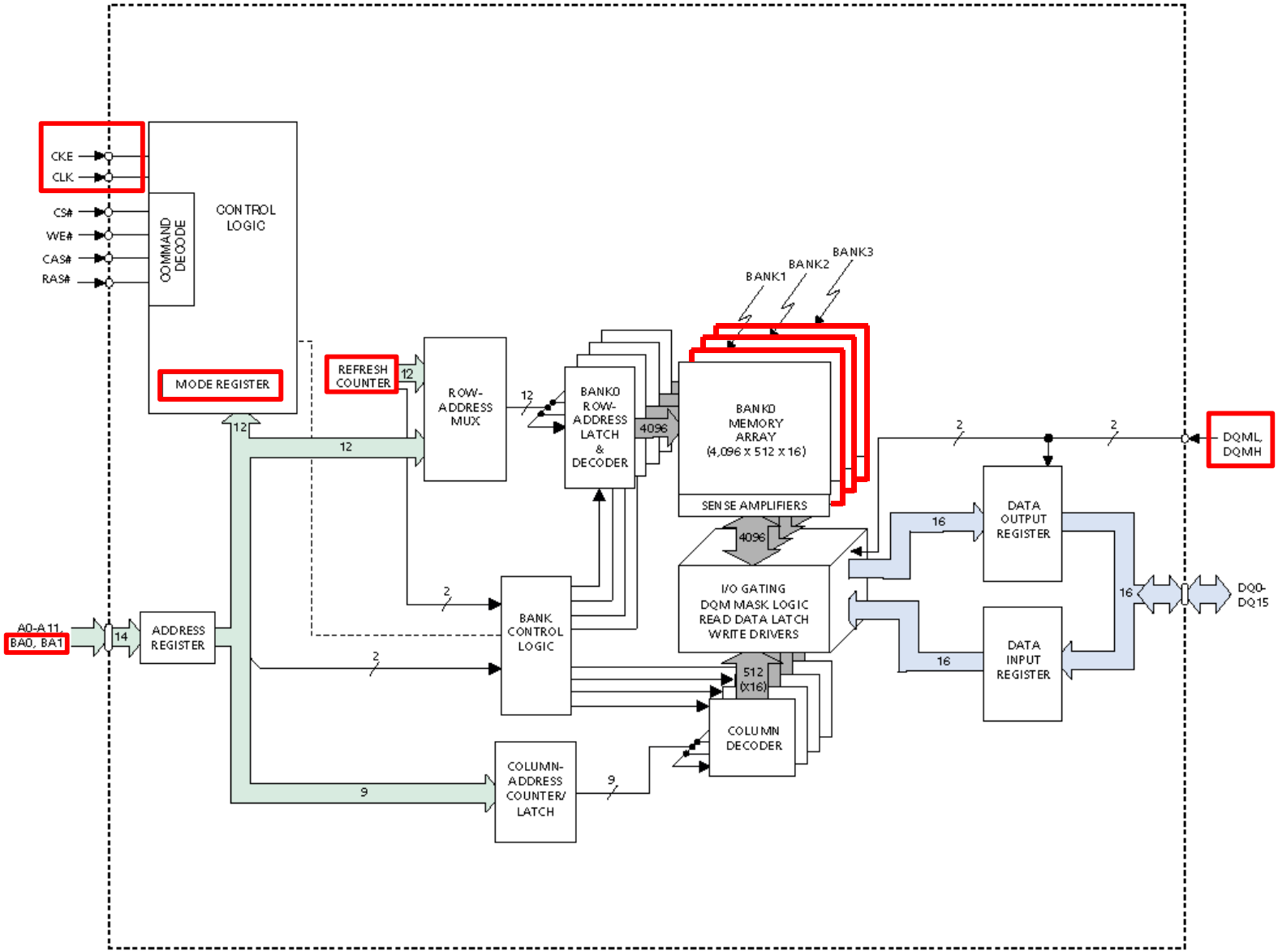
Memorias SDRAM

- Las SDRAM son una evolución de las DRAM con interfaz síncrono
  - Controlar de una manera robusta una SDRAM es más simple que hacerlo de una DRAM al no depender de la temporización absoluta de señales asíncronas.
- Adicionalmente, las SDRAM:
  - Se controlan mediante comandos (combinación de niveles de las señales de control).
  - Disponen de múltiples bancos (arrays de celdas):
    - Si se hace un acceso entrelazado a los mismos puede ocultarse el tiempo de precarga.
  - Disponen de 2 modos de refresco:
    - Auto-refresh: Usa un contador interno que evita direccionar externamente la SDRAM.
    - Self-Refresh: Refresca la memoria durante periodos de inactividad.
  - Tienen un registro de modo que permite programar:
    - La latencia (número de ciclos) desde el envío de un comando a la salida de datos.
    - El núm. de palabras (1, 2, 4, 8, fila completa) que pueden leerse/escribirse en una ráfaga.
  - Tras el encendido y antes de poder usarse, la SDRAM de ser inicializada
    - Tras la estabilización de reloj debe esperarse un cierto tiempo enviado NOP.
    - Deben precargarse todos los bancos del dispositivo.
    - Deben lanzarse 8 operaciones de Auto-refresco.
    - Debe programarse el registro de modo.

J.M. Mendías
2017

Memorias DRAM

estructura

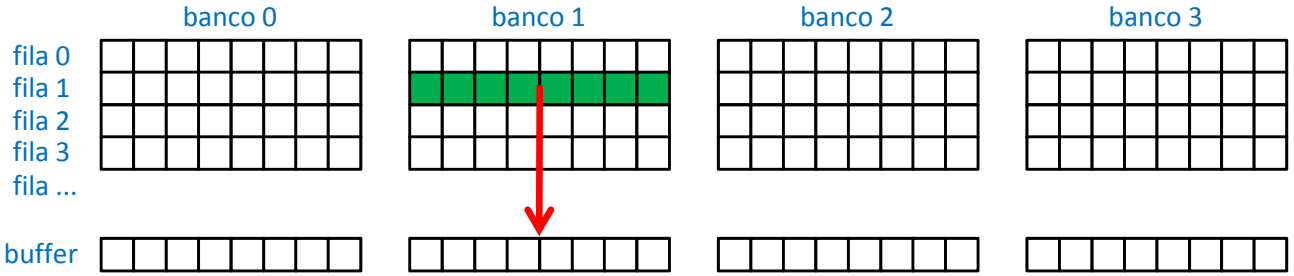


J.M. Mendías 2017

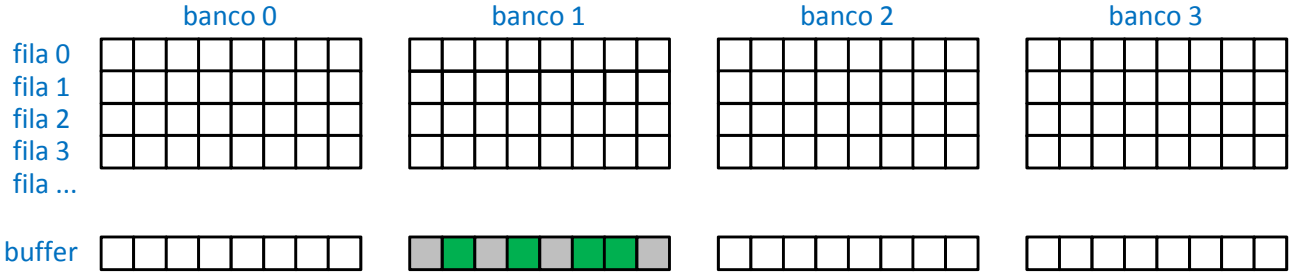
Memorias DRAM

ciclos de acceso (i)

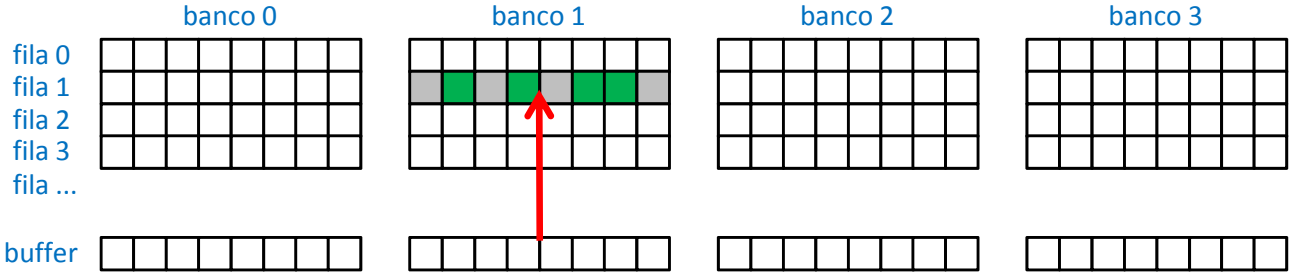
- Todo ciclo de acceso a datos de un banco repite las mismas operaciones:
  - Se activa una fila, copiándose al completo sobre el buffer de datos:



  - Se realizan operaciones de lectura/escritura en ráfaga sobre la fila activa (en el buffer):



  - La fila se precarga, copiándose el contenido del buffer sobre el array de celdas



J.M. Mendías
2017

Memorias DRAM

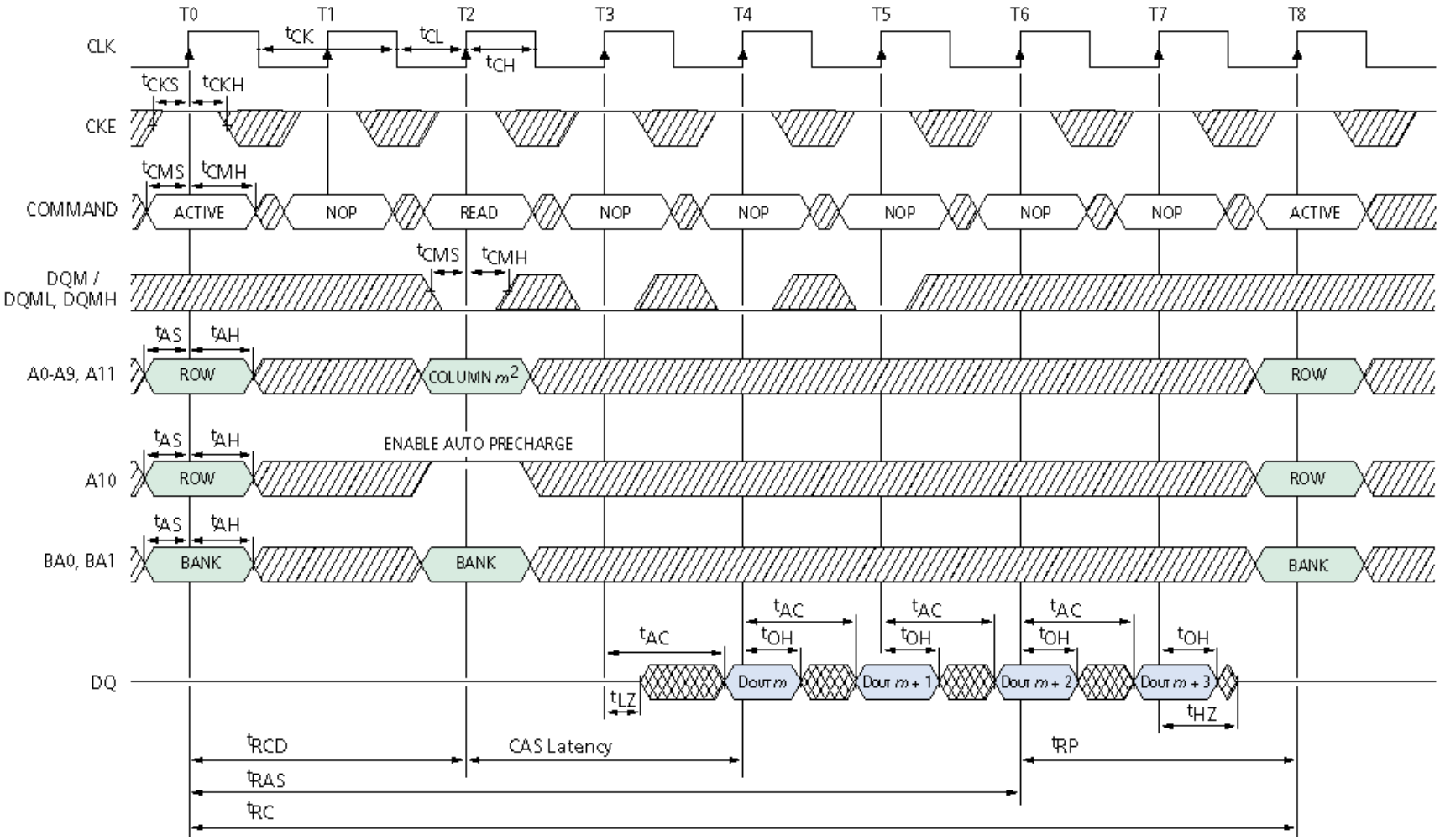
ciclos de acceso (ii)

- Los comandos más comunes que se envían a una SDRAM son:
  - NOP: Ignorar el resto de entradas
    - (CE, RAS, CAS, WE, DQ) = (0, 1, 1, 1, 0)
  - Activate: Activa una fila de un banco en particular
    - (CE, RAS, CAS, WE, DQ) = (0, 0, 1, 1, 0)
  - Read: Inicia una lectura en ráfaga de datos de la fila activa
    - (CE, RAS, CAS, WE, DQ) = (0, 1, 0, 1, 0)
    - Existe variaciones de con o sin autoprecarga al finalizar la ráfaga de lectura.
  - Write: Inicia una escritura en ráfaga de datos en la fila activa
    - (CE, RAS, CAS, WE, DQ) = (0, 1, 0, 0, 0)
    - Existe variaciones de con o sin autoprecarga al finalizar la ráfaga de escritura.
  - Precharge: Cierra una fila de un banco en particular
    - (CE, RAS, CAS, WE, DQ) = (0, 0, 1, 0, 1)
  - Refresh: Inicia una operación de refresco
    - (CE, RAS, CAS, WE, DQ) = (0, 0, 0, 1, 1)
  - Mode register set: Programa el registro de modo
    - (CE, RAS, CAS, WE, DQ) = (0, 0, 0, 0, 1)

Memorias SDRAM

ciclos de acceso (iii)

- Ciclo de lectura en ráfaga con auto precarga:

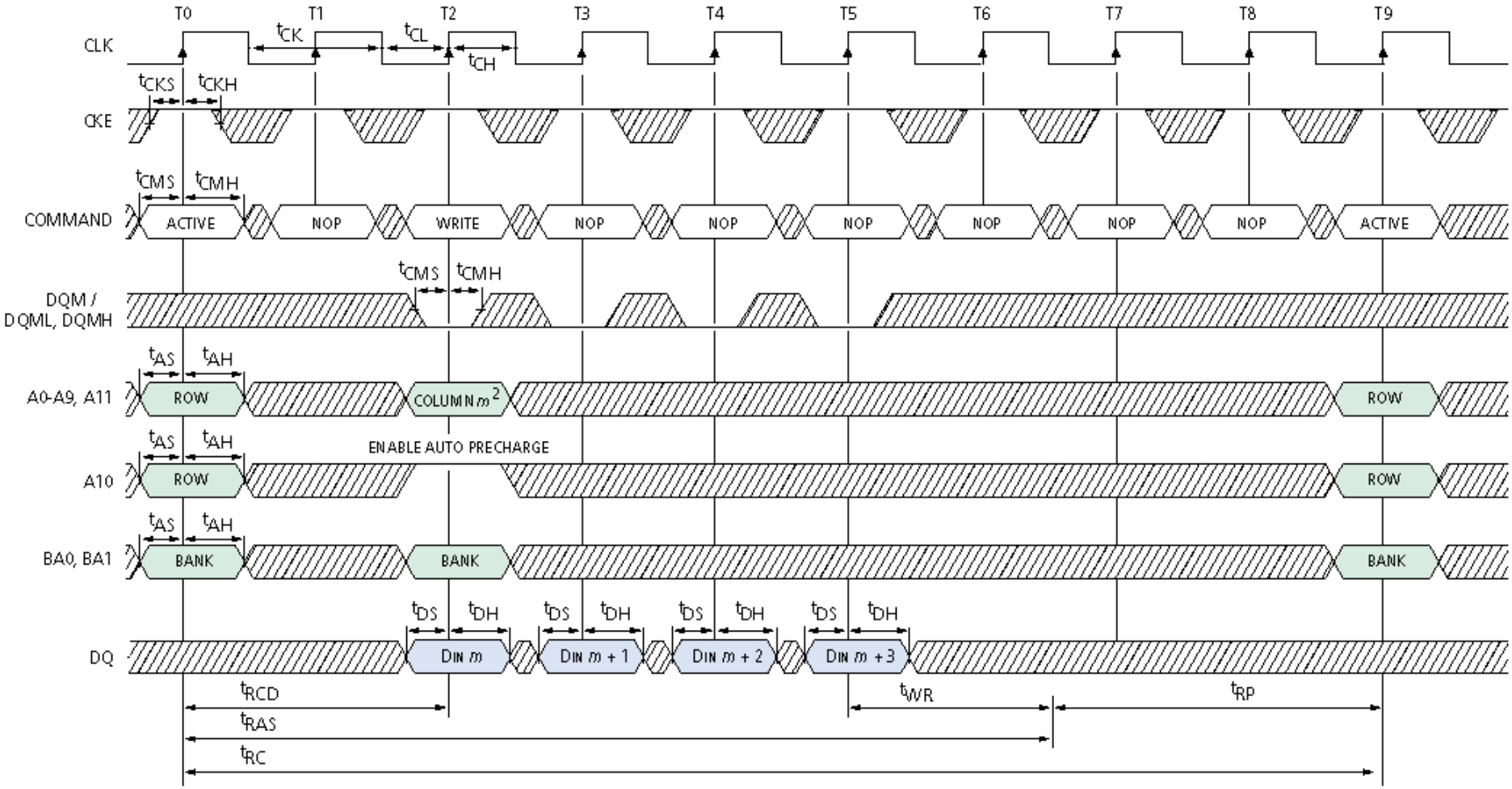


J.M. Mendías
2017

Memorias SDRAM

ciclos de acceso (iv)

- Ciclo de escritura en ráfaga con auto precarga:



J.M. Mendías 2017

Controlador de SDRAM

presentación

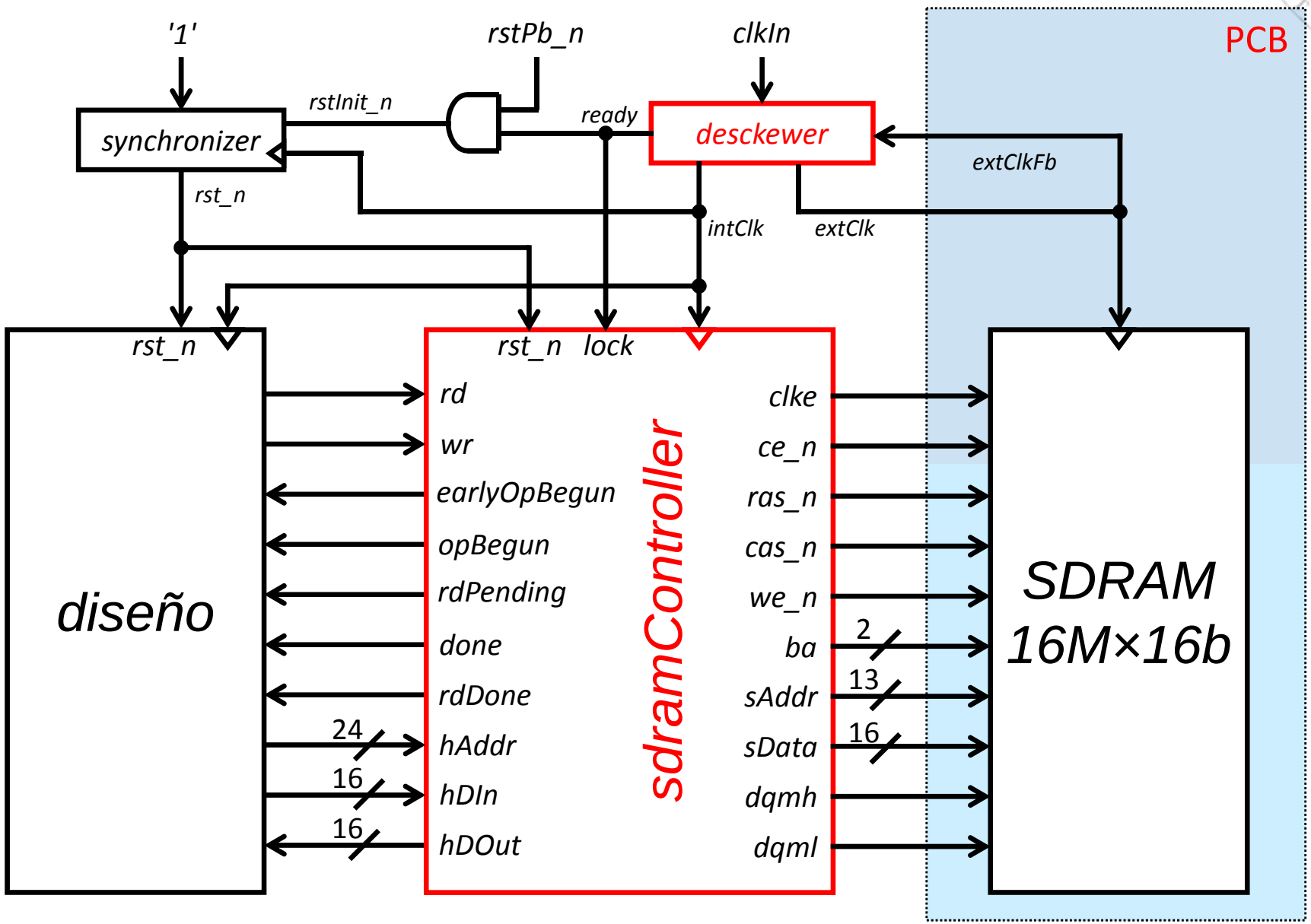
- Diseñar un controlador elemental de SDRAM que:
  - Ofrezca al host un bus de dirección no multiplexado de 24 bits (hAddr)
    - Ofrezca una vista de la SDRAM como una memoria lineal de 16M x 16b
    - Mapeando los bancos secuencialmente (se direccionan con los 2 MSB de hAddr)
  - Ofrezca al host buses separados para entrada/salida de datos (hDin/hDout)
    - Cada vez que rd/wr se active, comenzará la lectura/escritura en la SDRAM.
  - Dado que el tiempo de acceso a la SDRAM es variable dispondrá de señales para:
    - Indicar que una lectura/escritura/refresco ha comenzado (earlyOpBegun/opBegun)
    - Indicar que una lectura/escritura ha finalizado (done/rdDone)
    - Indicar que quedan operaciones de lectura pendientes (rdPending)
  - Inicialice la SDRAM y programe el MSR con:
    - 3 ciclos de CAS latency.
    - Ráfagas de 1 palabra (modo no-burst).
  - Refresque periódicamente la SDRAM (periodo de refresco 64 ms)
    - Pasándola a modo Self-refresh tras cierto periodo de inactividad.

- Dado que la SDRAM es externa, diseñar un eliminador de skew que
  - Corrija el desfase en la distribución de reloj por el PCB.

J.M. Mendías
2017

Controlador de SDRAM

aplicación al diseño



J.M. Mendías
2017

Controlador de SDRAM

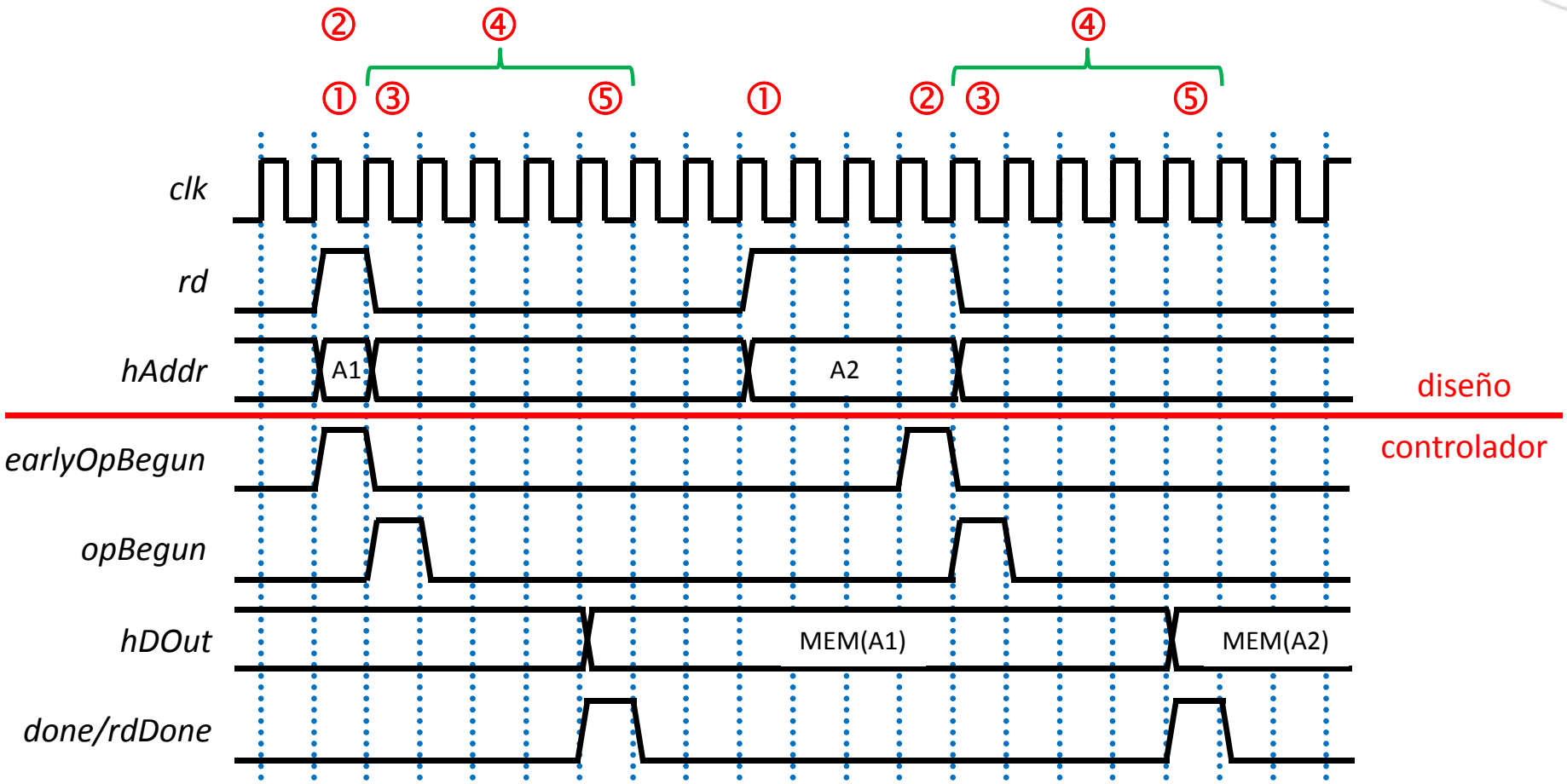
ciclos de acceso no segmentados

- Ciclo de lectura (no segmentada)
  1. Escribir dirección y activar rd
  2. Activa un ciclo la señal earlyOpBegun, la activación puede retrasarse si:
     - El controlador está refrescando una fila.
     - La dirección solicitada no se encuentra en el banco/fila activos.
  3. El controlador activa un ciclo la señal OpBegun e inicia la lectura en la SDRAM
  4. Desactivar rd, la dirección puede cambiar.
  5. El controlador carga el dato de la SDRAM y activa un ciclo la señal done y rdDone.
     - Si al ciclo siguiente rd estuviera aún activa, se iniciaría otro ciclo de lectura.
- Ciclo de escritura (no segmentada)
  1. Escribir dirección, dato y activar wr
  2. Activa un ciclo la señal earlyOpBegun, la activación puede retrasarse si:
     - El controlador está refrescando una fila.
     - La dirección solicitada no se encuentra en el banco/fila activos.
     - Se esta efectuando una operación de lectura previa
  3. El controlador activa un ciclo la señal OpBegun e inicia la escritura en la SDRAM
  4. Desactivar wr, la dirección puede cambiar.
     - Si al ciclo siguiente wr estuviera aún activa, se iniciaría otro ciclo de escritura.

J.M. Mendías
2017

Controlador de SDRAM

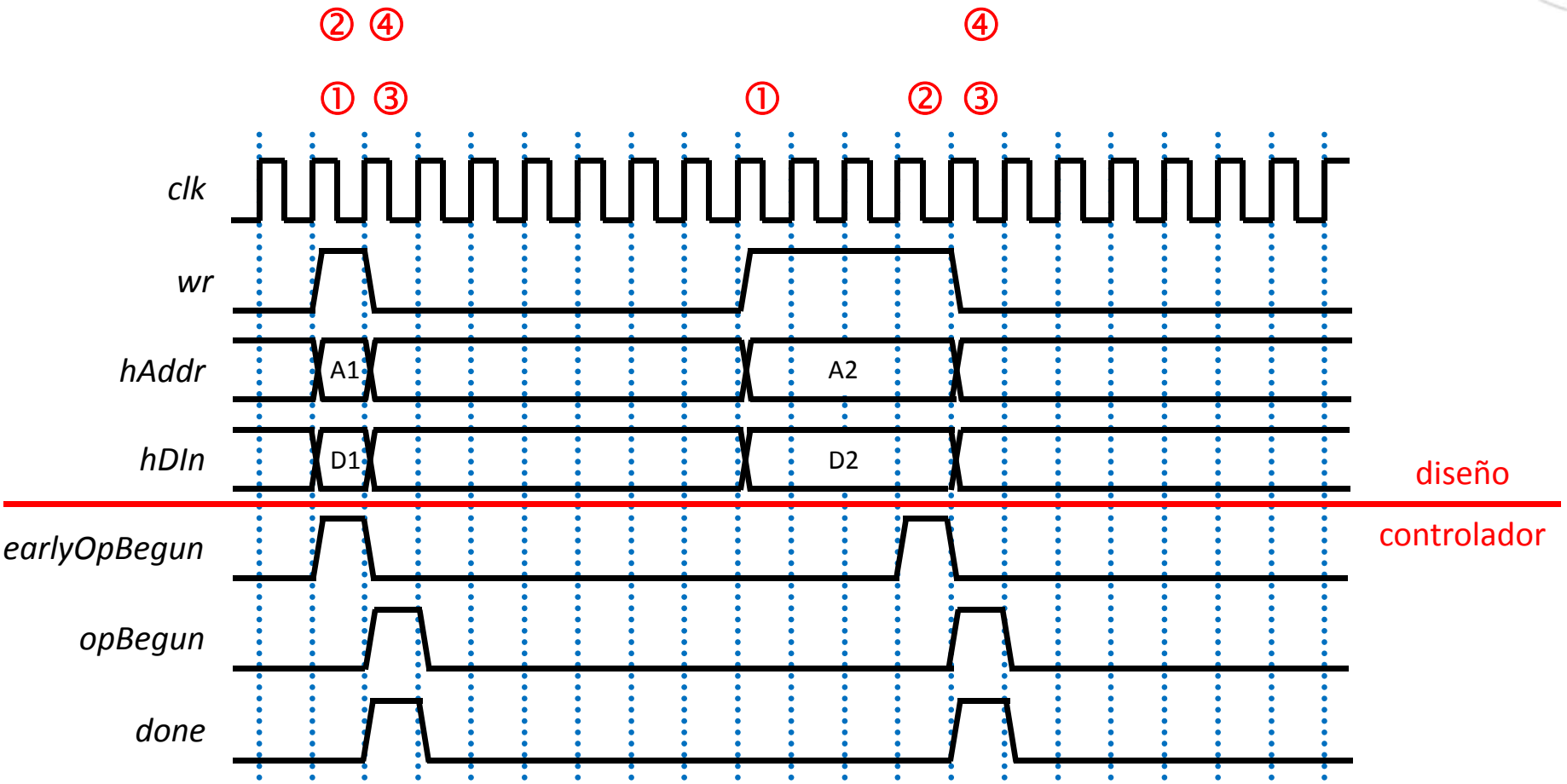
operación de lectura no segmentada



J.M. Mendías
2017

Controlador de SDRAM

operación de escritura no segmentada

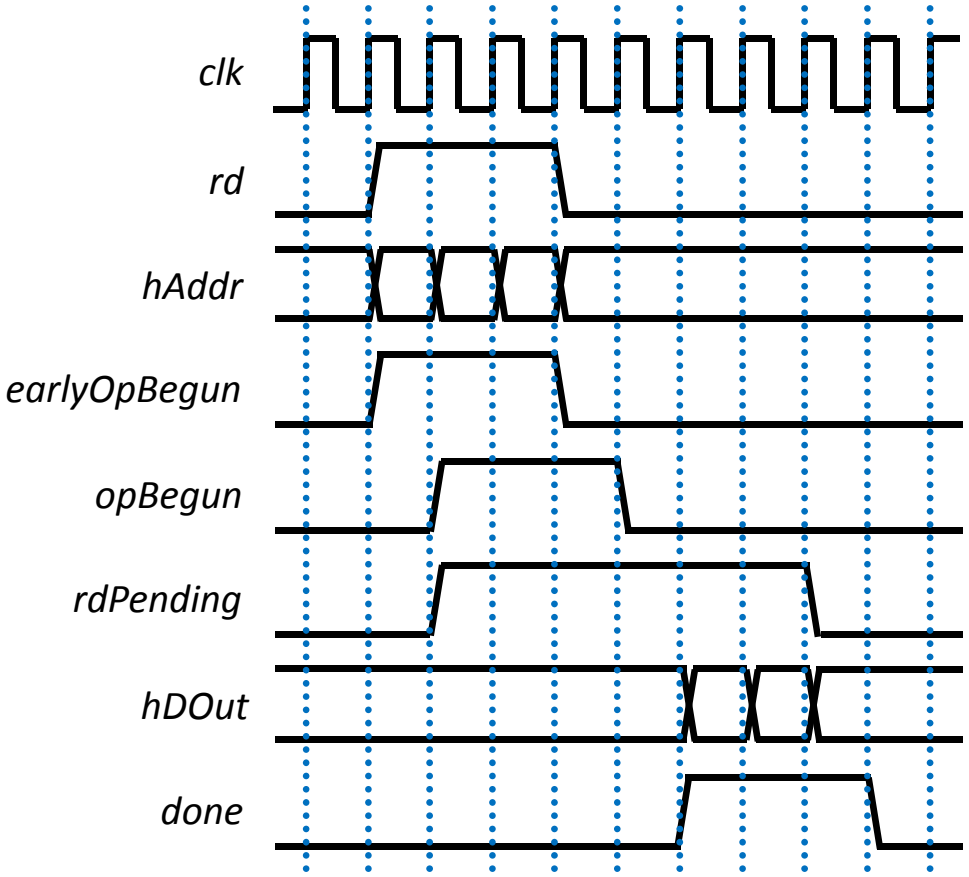


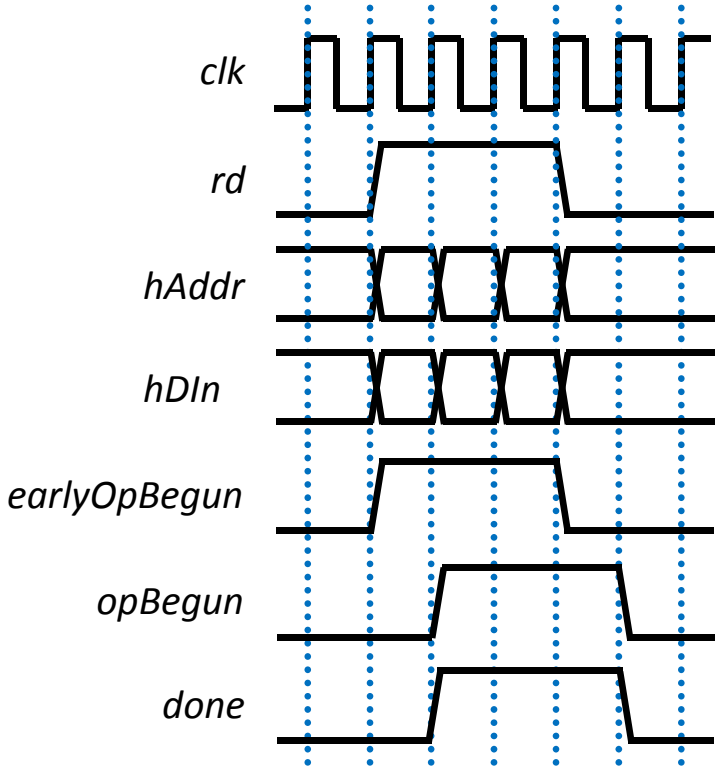
J.M. Mendías 2017

Controlador de SDRAM

ciclos de acceso segmentados

- Cuando las direcciones pertenecen a la misma fila, es posible lanzar una operación de lectura/escritura por ciclo (equivale a modo burst).
  - Las direcciones cambian ciclo a ciclo y manteniendo activas las señales rd/wr.





J.M. Mendías
2017

Controlador de SDRAM

configuración

- El controlador es parametrizable usando los genéricos siguientes :
  - **FREQ**: Frecuencia de operación
  - **PIPE_EN**: Habilita la segmentación de operaciones de lectura/escritura
  - **MAX_NOP**: Número máximo de NOP antes de que el controlador haga entrar a la SDRAM en modo Self-Refresh
  - **ENABLE_REFRESH**: Si cierto, inserta refrescos de fila cuando corresponda.
  - **MULTIPLE_ACTIVE_ROWS**: Si cierto, mantiene una línea activa por banco.
  - **DATA_WIDTH**: Anchura de datos.
  - **NROWS**: Número de filas de la SDRAM.
  - **NCOLS**: Número de columnas de la SDRAM.
  - **HADDR_WIDTH**: Anchura de la dirección en el lado del host.
  - **SADDR_WIDTH**: Anchura de la dirección en el lado de la SDRAM.

- Adicionalmente dentro de la arquitectura se definen como constantes los diferentes parámetros de temporización.

Controlador de SDRAM

estados

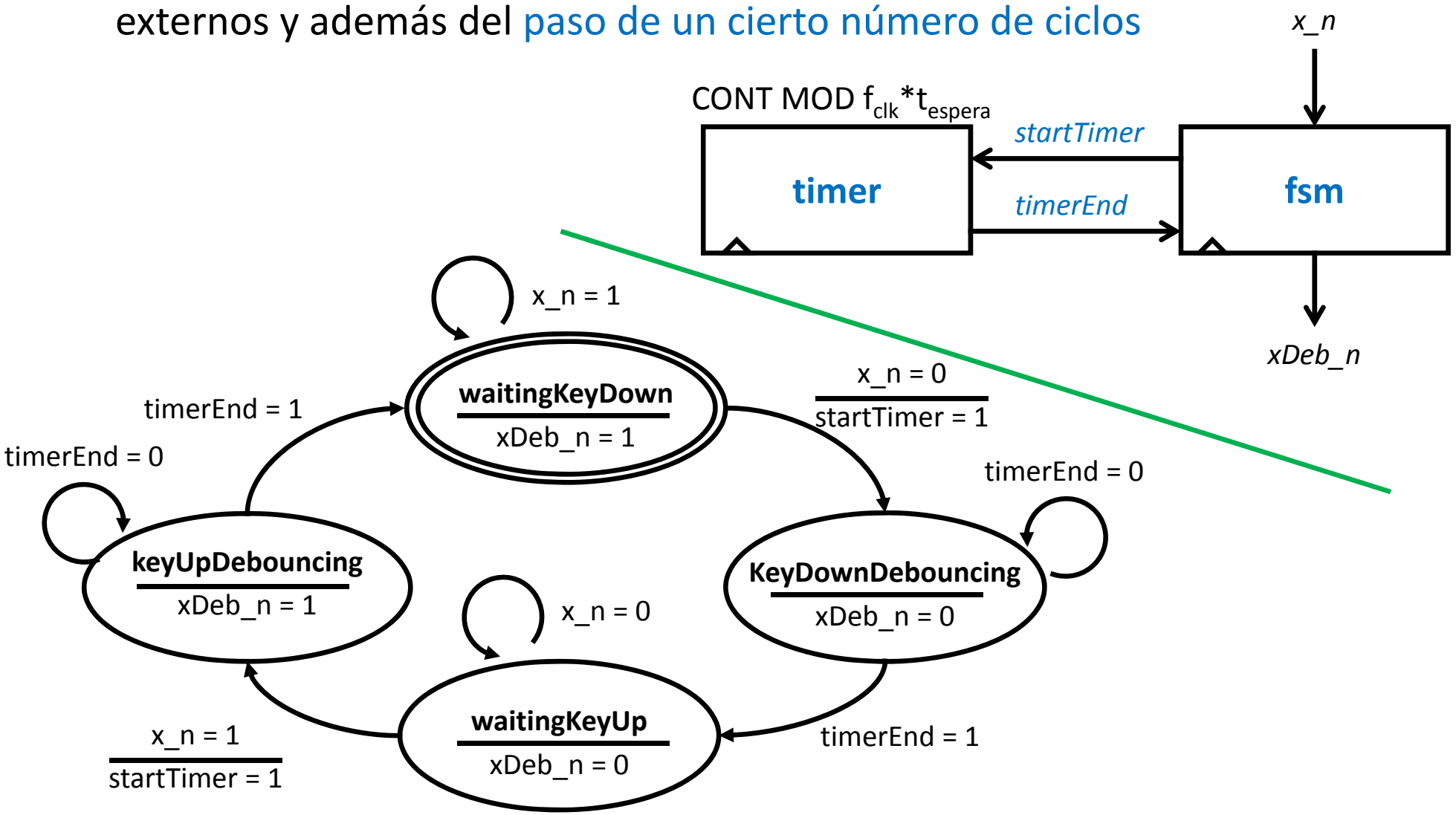
- El controlador puede estar en los siguientes estados:
  - **INITWAIT**: Espera a que finalice la inicialización de power-on
  - **INITPCHG**: Realiza una precarga inicial de los bancos de la SDRAM.
  - **INITRFSG**: Realiza los refrescos iniciales de la SDRAM
  - **INITSETMODE**: Programa el modo de funcionamiento de la SDRAM
  - **RW**: Lee/escribe/refresca la SDRAM
  - **ACTIVATE**: Abre una fila de la SDRAM para lectura/escritura
  - **REFRESHROW**: Refresca una fila de la SDRAM
  - **SELFREFRESH**: Mantiene la SDRAM en modo self-refresh

- Las transiciones de estados se realizan en función de las condiciones y del número de ciclos necesario.

FSM temporizadas

eliminador de rebotes revisitado (i)

- En ocasiones, el cambio de estado de una FSM depende de eventos externos y además del paso de un cierto número de ciclos



J.M. Mendías
2017

FSM temporizadas

debouncer.vhd

```
timer:
process (rst_n, clk)
  constant TIMEOUT : natural := (BOUNCE*FREQ)-1;
  variable count   : natural range 0 to TIMEOUT;
begin
  if count=0 then
    timerEnd <= '1';
  else
    timerEnd <= '0';
  end if;
  if rst_n='0' then
    count := 0;
  elsif rising_edge(clk) then
    if startTimer='1' then
      count := TIMEOUT ;
    elsif timerEnd='0' then
      count := count - 1;
    end if;
  end if;
end process;
```

KHz = ciclos/ms al multiplicarlo por ms resultan los ciclos que se necesita contar

la cuenta es local al proceso, es de tipo natural por comodidad. La herramienta determinará el número de bits necesarios

usa una variable porque la fsm no necesita conocer la cuenta, solo necesita saber el final de ella

J.M. Mendías
2017

FSM temporizadas

debouncer.vhd

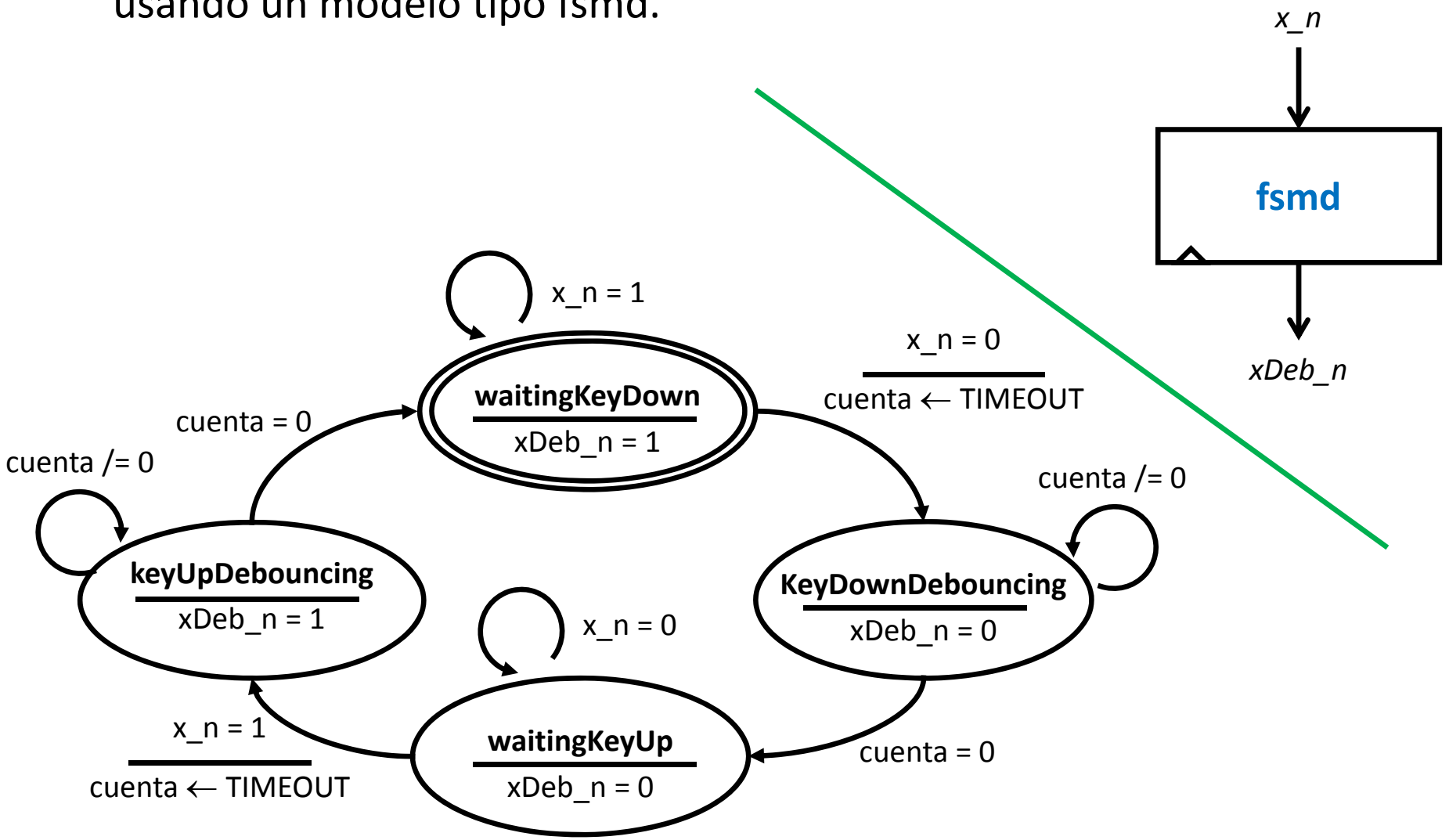
```
fsm:
process (rst_n, clk, x_n)
  type states is (
    waitingKeyDown,
    keyDownDebouncing,
    waitingKeyUp,
    KeyUpDebouncing);
  variable state: states;
begin
  xDeb_n <= '1';
  startTimer <= '0';
  case state is
    when waitingKeyDown =>
      if x_n='0' then
        startTimer <= '1';
      end if;
    when keyDownDebouncing =>
      xDeb_n <= '0';
    when waitingKeyUp =>
      xDeb_n <= '0';
      if x_n='1' then
        startTimer <= '1';
      end if;
    when KeyUpDebouncing =>
      null;
  end case;
...
```

```
...
  if rst_n='0' then
    state := waitingKeyDown;
  elsif rising_edge(clk) then
    case state is
      when waitingKeyDown =>
        if x_n='0' then
          state := keyDownDebouncing;
        end if;
      when keyDownDebouncing =>
        if timerEnd='1' then
          state := waitingKeyUp;
        end if;
      when waitingKeyUp =>
        if x_n='1' then
          state := KeyUpDebouncing;
        end if;
      when KeyUpDebouncing =>
        if timerEnd='1' then
          state := waitingKeyDown;
        end if;
    end case;
  end if;
end process;
```

FSM temporizadas

eliminador de rebotes revisitado (ii)

- En estos casos, puede integrarse el timer dentro de la propia fsm usando un modelo tipo fsmd.



J.M. Mendías
2017

FSM temporizadas

debouncer-fsmd.vhd

```
fsmd:
process (rst_n, clk, x_n)

  constant TIMEOUT : natural
    := (BOUNCE*FREQ)-1;
  type states is (
    waitingKeyDown,
    keyDownDebouncing,
    waitingKeyUp,
    KeyUpDebouncing);
  variable state : states;
  variable count : natural
    range 0 to TIMEOUT;

begin

  xDeb_n <= '1';
  if state=keyDownDebouncing
    or state=waitingKeyUp then
    xDeb_n <= '0';
  end if;
...
```

solo cambia de estado si el temporizador a finalizado la cuenta de ciclos

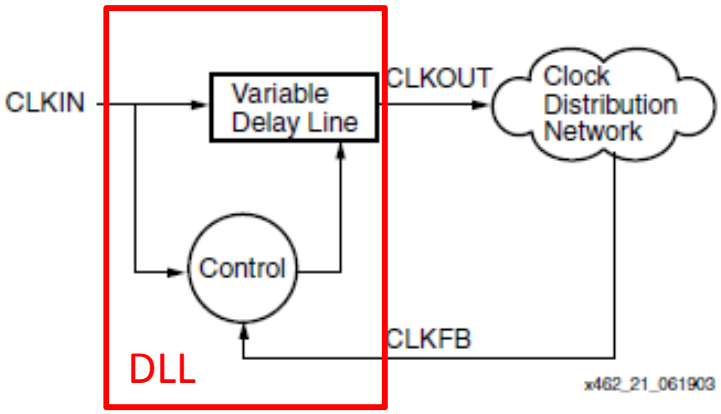
programa el temporizador (el número de ciclos puede ser distinto en cada estado)

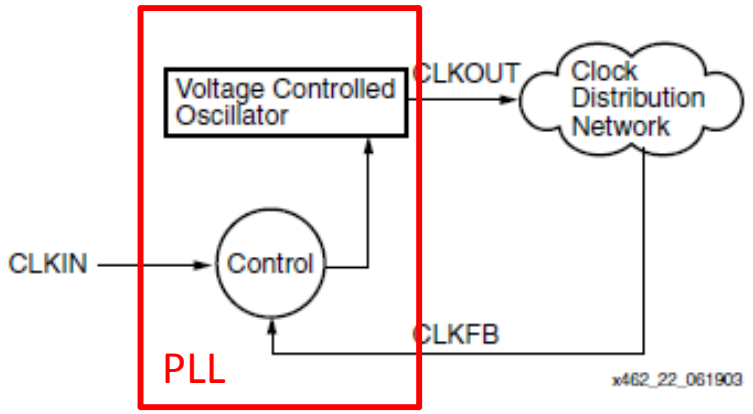
```
...
  if rst_n='0' then
    state := waitingKeyDown;
    count := 0;
  elsif rising_edge(clk) then
    if count/=0 then
      count := count-1;
    else
      case state is
        when waitingKeyDown =>
          if x_n='0' then
            state := keyDownDebouncing;
            count := TIMEOUT;
          end if;
        when keyDownDebouncing =>
          state := waitingKeyUp;
        when waitingKeyUp =>
          if x_n='1' then
            state := KeyUpDebouncing;
            count := TIMEOUT;
          end if;
        when KeyUpDebouncing =>
          state := waitingKeyDown;
      end case;
    end if;
  end if;

end process;
```

Señal de reloj

eliminación del skew

- Cuando 2 sistemas síncronos se comunican, existe problema de skew.
  - El desfase en la llegada de reloj a los distintos sistemas puede provocar que la comunicación no sea fiable.
- El problema es mayor si los sistemas están integrados en distintos chip.
  - Por ejemplo, a partir de 25 MHz, la comunicación entre controlador (FPGA) – SDRAM es imposible si no se corrige el skew.
- Para corregir el skew, se retarda la señal de reloj distribuida hasta que esté en fase con la señal de reloj de entrada:
  - Delay-Locked Loop (DLL): Línea de retardo variable + lógica de control.
  - Phase-Locked Loop (PLL): Oscilador regulable + lógica de control.

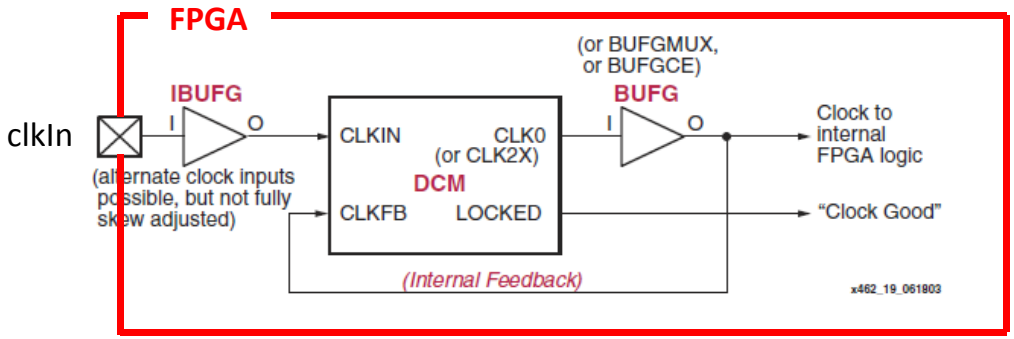




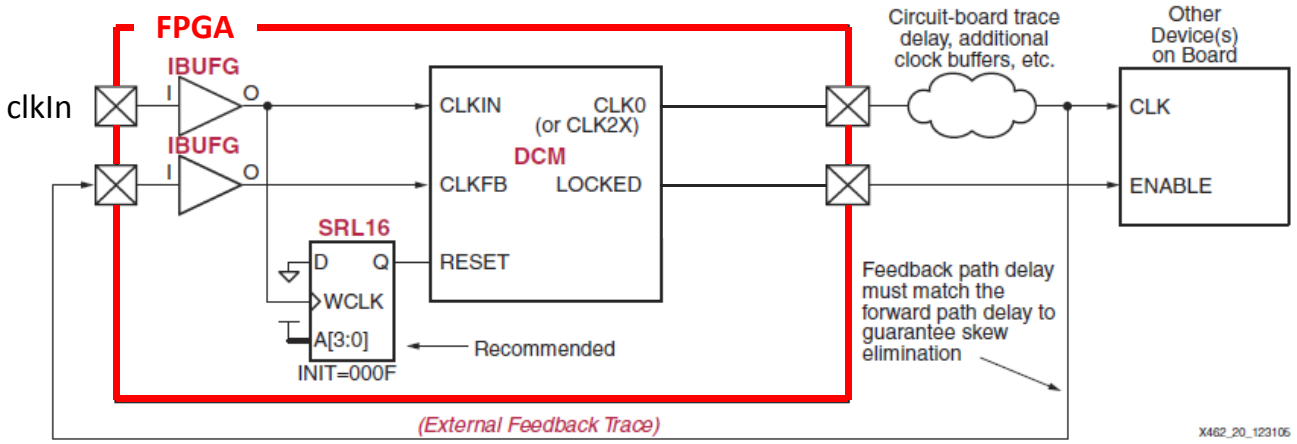
Digital Clock Manager

eliminación del skew

- El DCM de la FPGA puede usarse para eliminar de skew interno
  - Derivado de la distribución de la señal de reloj dentro de la FPGA



- También puede usarse para eliminar del skew externo
  - Derivado de la distribución de la señal de reloj dentro de la placa PCB



J.M. Mendías
2017

Señal de reloj

deskewer.vhd

```
library ieee;
use ieee.std_logic_1164.all;

entity desckewer is
  generic (
    FREQ     : natural           -- frecuencia del reloj de entrada en KHz
  );
  port (
    clkIn    : in  std_logic;    -- oscilador externo
    ready    : out std_logic;    -- se activa cuando las señales de reloj son validas
    intClk   : out std_logic;    -- reloj interno con bajo skew
    extClk   : out std_logic;    -- reloj externo con bajo skew
    extClkFb : in  std_logic     -- señal de feedback del reloj externo
);
end desckewer;

library unisim;
use unisim.vcomponents.all;

architecture syn of desckewer is

  signal clkIn_b, clkfb, clk0 : std_logic;
  signal extClkFb_b : std_logic;
  signal reset : std_logic;
  signal intRdy, extRdy : std_logic;

begin
  ...
end syn;
```

Señal de reloj

deskewer.vhd

```
clkInBuffer : IBUFG
 port map ( I => clkIn, O => clkIn_b );

clkFbBuffer : BUFG
 port map ( I => clk0, O => clkFb );

intClk <= clkFb;

intClkDesckewer : DCM
 generic map
 (
   CLKDV_DIVIDE          => 2.0,
   CLKFX_DIVIDE          => 2,
   CLKFX_MULTIPLY        => 2,
   CLKIN_DIVIDE_BY_2     => FALSE,
   CLKIN_PERIOD          => 1_000_000.0/real(FREQ),
   CLKOUT_PHASE_SHIFT    => "NONE",
   CLK_FEEDBACK          => "1X",
   DESKEW_ADJUST         => "SYSTEM_SYNCHRONOUS",
   DFS_FREQUENCY_MODE    => "LOW",
   DUTY_CYCLE_CORRECTION => FALSE,
   PHASE_SHIFT           => 0,
   STARTUP_WAIT          => FALSE
 )
```

```
port map
 (
   CLK0     => clk0,
   CLK90    => open,
   CLK180   => open,
   CLK270   => open,
   CLK2X    => open,
   CLK2X180 => open,
   CLKDV    => open,
   CLKFX    => open,
   CLKFX180 => open,
   LOCKED   => intRdy,
   PSDONE   => open,
   STATUS   => open,
   CLKFB    => clkFb,
   CLKIN    => clkIn_b,
   PSCLK    => '0',
   PSEN     => '0',
   PSINCDEC => '0',
   RST      => '0'
 );
```

eliminador de skew interno

J.M. Mendías 2017

Señal de reloj

deskewer.vhd

```
extClkFbBuffer : IBUFG
  port map ( I => extClkFb, O => extClkFb_b );

resetGenerator : SRL16
 generic map ( INIT => X"000F" )
 port map ( CLK => clkIn_b, A0 => '1', A1 => '1', A2 => '1', A3 => '1', D => '0', Q => reset );

ready <= intRdy and extRdy;

extclkDesckewer : DCM
 generic map
 (
   CLKDV_DIVIDE          => 2.0,
   CLKFX_DIVIDE          => 2,
   CLKFX_MULTIPLY        => 2,
   CLKIN_DIVIDE_BY_2     => FALSE,
   CLKIN_PERIOD          => 1_000_000.0/real(FREQ),
   CLKOUT_PHASE_SHIFT    => "NONE",
   CLK_FEEDBACK          => "1X",
   DESKEW_ADJUST         => "SYSTEM_SYNCHRONOUS",
   DFS_FREQUENCY_MODE    => "LOW",
   DUTY_CYCLE_CORRECTION => FALSE,
   PHASE_SHIFT           => 0,
   STARTUP_WAIT          => FALSE
 )
```

```
port map
 (
   CLK0     => extClk,
   CLK90    => open,
   CLK180   => open,
   CLK270   => open,
   CLK2X    => open,
   CLK2X180 => open,
   CLKDV    => open,
   CLKFX    => open,
   CLKFX180 => open,
   LOCKED   => extRdy,
   PSDONE   => open,
   STATUS   => open,
   CLKFB    => extClkFb_b,
   CLKIN    => clkIn_b,
   PSCLK    => '0',
   PSEN     => '0',
   PSINCDEC => '0',
   RST      => reset
 );
```

eliminador de skew externo

Diseño principal

diseño RTL

- El diseño deberá constar de:
  - Un iisInterface que reciba y transmita las muestras.
  - Un sdramController que almacene las muestras grabadas.
    - Será direccionado por 2 punteros uno para reproducción y otro para grabación.
  - Un contador que almacene el volumen actual.
    - Se inc/decrementarán a presiones de los respectivos pulsadores.
  - Un desplazador combinacional que reduzca el volumen de la muestras
    - Desplazando a derechas la muestra enviada al iisInterface un número de bits equivalente al volumen actual.
    - Recuérdese que las muestras están codificadas en C2.
  - Un contador BCD de segundos transcurridos.
    - Se inc/decrementará cada segundo cuando grave/reproduzca.
  - Una FDMD con los siguientes estados:
    - initial: Espera la pulsación de rec_n
    - recording: Grabando hasta la pulsación de rec_n o el transcurso de 60s
    - writeSample: Escribe una muestra en la memoria (se alterna con recording)
    - waiting: Espera la pulsación de rec_n o paly_n
    - playing: Reproduciendo hasta la pulsación de rec_n o el fin de la grabación.
    - readSample: Lee una muestra en la memoria (se alterna con playing)

Diseño principal
consideraciones

- El sistema debe grabar como máximo 60s de sonido, que ocupa:
  - (duración) × f_s × (num. canales) × (anchura de datos) ÷ 8
  - 60 × 48.800 × 2 × 16 ÷ 8 ≈ 11,2 MB < 32 MB SDRAM

- El controlador de SDRAM tarda en hacer un acceso
  - Entre 5 y 10 ciclos (sin ráfaga)
    - Si la fila está activa una lectura/escritura tarda: 5 ciclos
    - Se hay que activar la fila se añaden: 1 ciclo de precarga + 1 ciclo de activación
    - El controlador refresca una fila cada (64 ms) ÷ (8192 filas) × 50MHz ≈ 390 ciclos
    - Cada refresco de fila tarda: (66 ns) × (50MHz) ≈ 3 ciclos

- El audio CODEC muestrea a 48,8 KHz
  - inSampleRdy y outSampleRqt se activan cada 512 ciclos, pero:
    - Desde una activación de outSampleRqt (lectura de SDRAM) hasta una de inSampleRdy (escritura en SDRAM) pasan 128 ciclos < 10 ciclos
    - Desde una activación de inSampleRdy (escritura de SDRAM) hasta una de outSampleRqt (lectura de SDRAM) pasan 384 ciclos < 10 ciclos

Tareas

1. Crear el proyecto **`lab9`** en el directorio **`DAS`**
2. Descargar de la Web en el directorio **`common`** los ficheros
   - **`iisInterface.vhd`**, **`deskewer.vhd`** y **`sdramController.vhd`**
3. Descargar de la Web en el directorio **`lab9`** los ficheros:
   - **`lab9.vhd`** y **`lab9.ucf`**
4. Completar el fichero **`common.vhd`** con la declaración de los nuevos componentes reusables.
5. Completar el código omitido en los ficheros:
   - **`iisInterface.vhd`** y **`lab9.vhd`**
6. Añadir al proyecto los ficheros:
   - **`common.vhd`**, **`synchronizer.vhd`**, **`debouncer.vhd`**, **`edgeDetector.vhd`**, **`bin2segs.vhd`**, **`iisInterface.vhd`**, **`deskewer.vhd`**, **`sdramController.vhd`**, **`lab9.vhd`** y **`lab9.ucf`**
7. Sintetizar, implementar y generar el fichero de configuración.
8. Conectar el cable de audio, unos auriculares, arrancar la reproducción de audio en el PC y encender la placa.
9. Descargar el fichero **`lab9.bit`**

Acerca de *Creative Commons*

- Licencia CC (Creative Commons)

  - Ofrece algunos derechos a terceras personas bajo ciertas condiciones. Este documento tiene establecidas las siguientes:

Reconocimiento (*Attribution*):
En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.

No comercial (*Non commercial*):
La explotación de la obra queda limitada a usos no comerciales.

Compartir igual (*Share alike*):
La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Más información: https://creativecommons.org/licenses/by-nc-sa/4.0/