Diseño Digital

2024-2

Prof. Alonso Sanchez Huapaya

pcelasan@upc.edu.pe

Preguntas

pcelasan@upc.edu.pe

Sobre la clase pasada…

- ¿Qué sentencias VHDL usamos para declarar un codificador/decodificador?

- ¿Qué sentencia VHDL usamos para declarar un multiplexor?

- ¿Cómo describimos la operación de SR/SL en VHDL?

pcelasan@upc.edu.pe

Temario

- Diseño síncrono
- Sentencias concurrentes y sentencias secuenciales
- Bloques digitales secuenciales básicos: FF D, registro
- Contadores
- Divisores de frecuencia

pcelasan@upc.edu.pe

Diseño síncrono

- Síncrono: que responde a una señal de reloj (CLK) // que cambia con los flancos de una señal de reloj (CLK).
  - Flip-flops
  - Contadores
  - Registros de desplazamiento
  - Otros – máquinas de estado, PWM, CPU, memorias, etc…

- Asíncrono: que no responde a / es independiente de una señal de reloj
  - Circuitos combinacionales

pcelasan@upc.edu.pe

Diseño síncrono

- “Reglas del diseño síncrono”: ¿Por qué existen?

1. Necesidad de sincronismo entre componentes secuenciales dentro de un mismo sistema digital (todos deben obedecer al mismo clock).

2. Existencia de la CDN (*Clock Distribution Network*): Componente circuital que se encarga de distribuir una señal de clock a través de un circuito integrado.

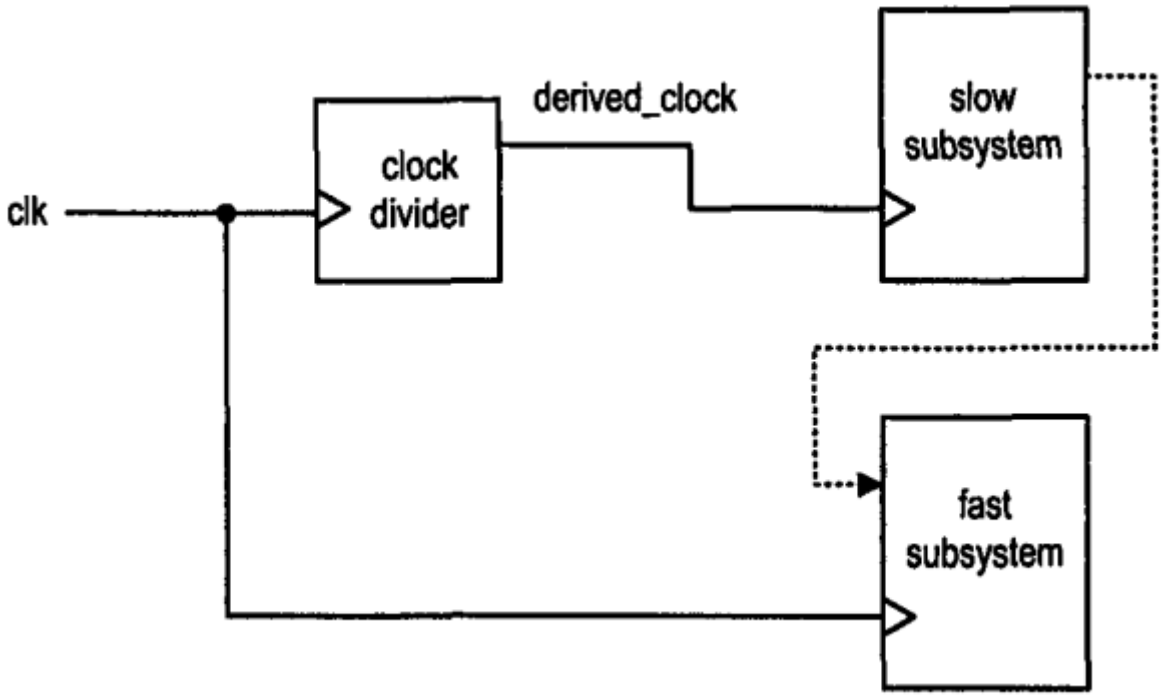
pcelasan@upc.edu.pe

Diseño síncrono

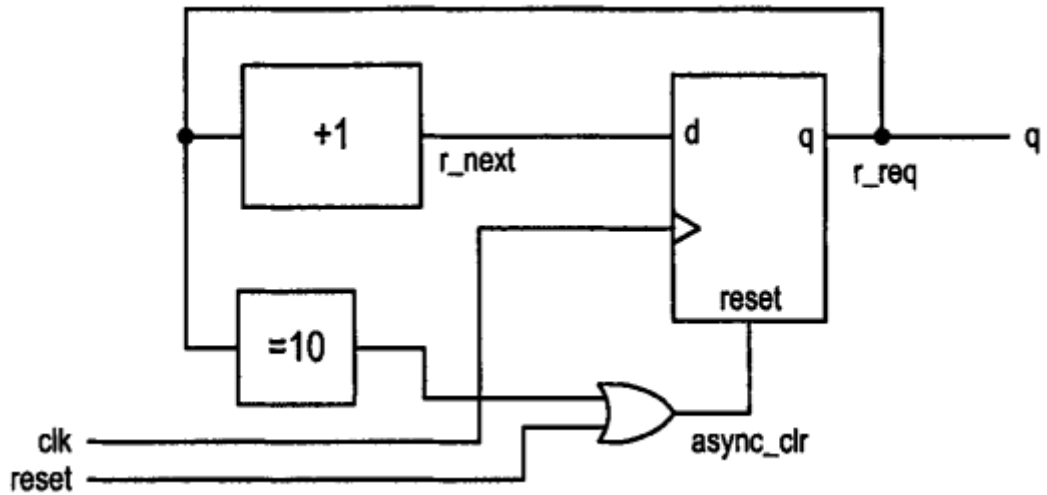
- “Reglas del diseño síncrono”:

1. Se usará un único CLK en todo tu sistema (no clocks derivados).
2. Cualquier entrada asíncrona se “muestreará” al ingresar al sistema.
3. Nunca se hará pasar la señal de CLK por una compuerta (*clock gating*).

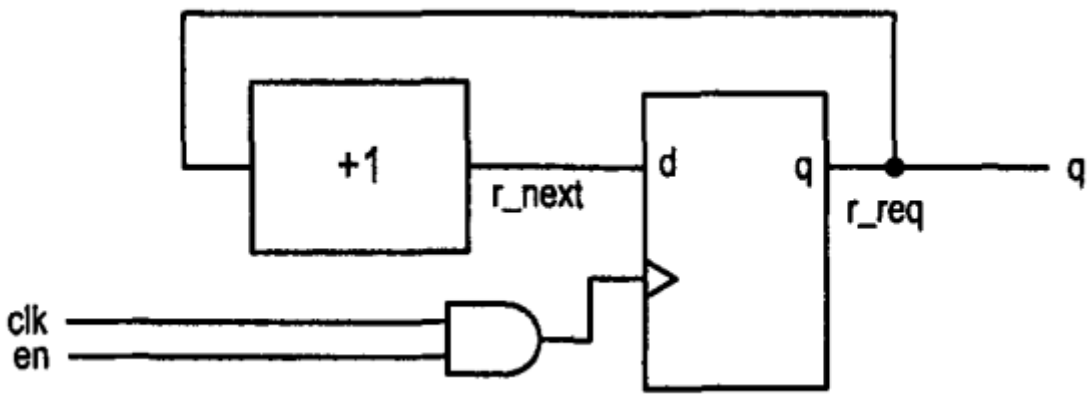
pcelasan@upc.edu.pe



Uso de clocks derivados



Uso de entradas asíncronas



Clock gating

pcelasan@upc.edu.pe

Sentencias concurrentes y sentencias secuenciales

- Concurrentes:
  - Asignación simple. Ejemplo: abc <= x(0) and x(1) or x(2)
  - Asignación condicional. Ejemplo: sentencia when…else…
  - Asignación selectiva. Ejemplo: sentencia with…select…

```
architecture cond_arch of decoder4 is
begin
    x <= "0001" when (s="00") else
         "0010" when (s="01") else
         "0100" when (s="10") else
         "1000";
end cond_arch;
```

```
architecture sel_arch of mux4 is
begin
   with s select
     x <= a when "00",
          b when "01",
          c when "10",
          d when others;
end sel_arch;
```

pcelasan@upc.edu.pe

Sentencias concurrentes y sentencias secuenciales

- Secuenciales:
  - Emplean la construcción “process”, **dentro del cual** se describen acciones a ejecutarse secuencialmente.
  - “Sentencia concurrente que describe el comportamiento de un circuito como si fuese una caja negra” (Chu, pág. 97).
  - Los “process” se deben ubicar dentro del cuerpo de la arquitectura.

```
process(sensitivity_list)
   declarations;
begin
   sequential statement;
   sequential statement;
   . . .
end process;
```

pcelasan@upc.edu.pe

Bloques digitales secuenciales básicos

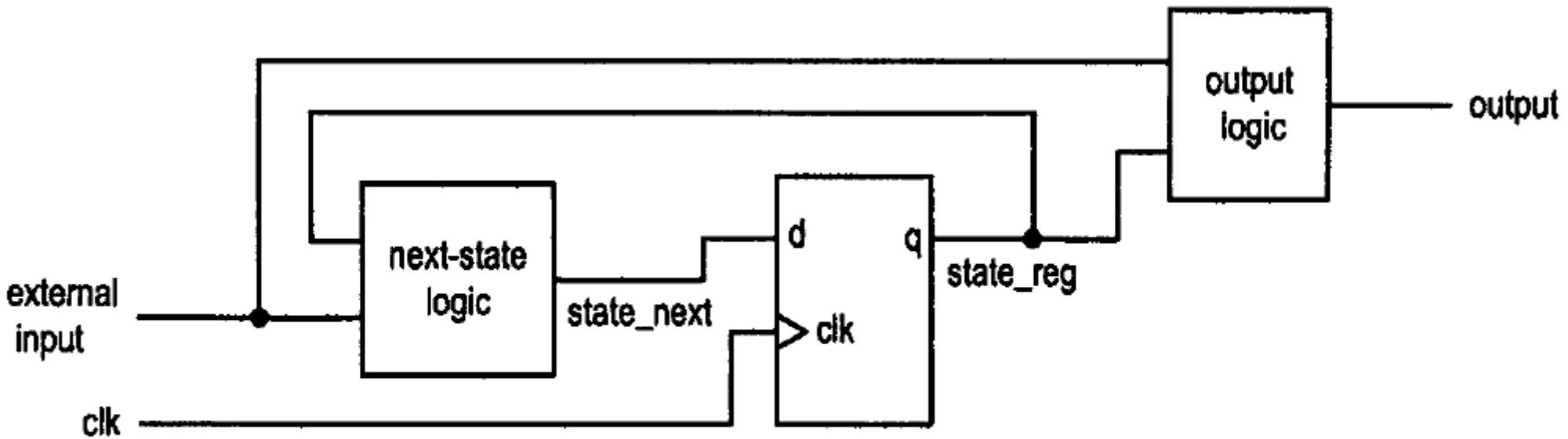


Figure 8.5 Conceptual diagram of a synchronous sequential circuit.

pcelasan@upc.edu.pe

Bloques digitales secuenciales básicos

- FF tipo D con RST (reset)
- Agregar: RST (reset) y ENA (habilitador)
- RTL en pizarra

```
library ieee;
use ieee.std_logic_1164.all;
entity dffr is
   port(
      clk: in std_logic;
      reset: in std_logic;
      d: in std_logic;
      q: out std_logic
   );
end dffr;

architecture arch of dffr is
begin
   process(clk,reset)
   begin
      if (reset='1') then
         q <='0';
      elsif (clk'event and clk='1') then
         q <= d;
      end if;
   end process;
end arch;
```

pcelasan@upc.edu.pe

Bloques digitales secuenciales básicos

- Registro (paralelo-paralelo): ejemplo de 8 bits

- RTL en pizarra

```
library ieee;
use ieee.std_logic_1164.all;
entity reg8 is
    port(
        clk: in std_logic;
        reset: in std_logic;
        d: in std_logic_vector(7 downto 0);
        q: out std_logic_vector(7 downto 0)
    );
end reg8;

architecture arch of reg8 is
begin
    process(clk,reset)
    begin
        if (reset='1') then
            q <=(others=>'0');
        elsif (clk'event and clk='1') then
            q <= d;
        end if;
    end process;
end arch;
```

pcelasan@upc.edu.pe

Tipo de datos dentro de numeric_std

- Antes de definir a los contadores, se definen los objetos de tipo “signed” o “unsigned”.

Importar: “numeric_std” (*use ieee.numeric_std.all;*)

Los “port” (in/out) siempre serán std_logic ó std_logic_vector
Sólo declararemos las “signal” como “signed” ó “unsigned”.
Hay que convertir signed/unsigned a std_logic para conectar…

pcelasan@upc.edu.pe

Tipo de datos dentro de numeric_std

- **Ejemplo 1:** Entidad con 1 entrada de 10 bits *ejm1_in* y 1 salida de 10 bits *ejm1_out*.

  La entidad realiza lo siguiente: ejm1_out = ejm1_in + ejm1_in.

  Tanto entrada como salida son “std_logic_vector” y representan enteros CON signo.

IMPORTANTE: El significado/interpretación de la cadena de bits lo da el desarrollador!

pcelasan@upc.edu.pe

Tipo de datos dentro de numeric_std

- **Ejemplo 2:** Entidad con 2 entradas de 10 bits cada una *a, b* y 1 salida de 10 bits *ejm2_out*.

  La entidad realiza lo siguiente: ejm2_out = a – b.

  Tanto entradas como salida son “std_logic_vector” y representan enteros CON signo.

IMPORTANTE: El significado/interpretación de la cadena de bits lo da el desarrollador!

pcelasan@upc.edu.pe

Contadores síncronos

- En VHDL…

- Diagrama RTL

- Diagrama de tiempos

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity binary_counter4_pulse is
   port(
      clk, reset: in std_logic;
      max_pulse: out std_logic;
      q: out std_logic_vector(3 downto 0)
   );
end binary_counter4_pulse;

architecture two_seg_arch of binary_counter4_pulse is
   signal r_reg: unsigned(3 downto 0);
   signal r_next: unsigned(3 downto 0);
begin
   -- register
   process(clk,reset)
   begin
      if (reset='1') then
         r_reg <= (others=>'0');
      elsif (clk'event and clk='1') then
         r_reg <= r_next;
      end if;
   end process;
   -- next-state logic (incrementor)
   r_next <= r_reg + 1;
   -- output logic
   q <= std_logic_vector(r_reg);
   max_pulse <= '1' when r_reg="1111" else
                '0';
end two_seg_arch;
```

pcelasan@upc.edu.pe

Divisores de frecuencia

- En VHDL…

- Diagrama RTL

- Diagrama de tiempos

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity binary_counter4_pulse is
   port(
      clk, reset: in std_logic;
      max_puls
      q: out s
   );
end binary_cou

architecture t                                    ulse is
   signal r_re
   signal r_ne
begin
   -- register
   process(clk
   begin
      if (rese
         r_reg
      elsif (c
         r_reg
      end if;
   end process;
   -- next-state logic (incrementor)
   r_next <= r_reg + 1;
   -- output logic
   q <= std_logic_vector(r_reg);
   max_pulse <= '1' when r_reg="1111" else
                '0';
end two_seg_arch;
```

pcelasan@upc.edu.pe