

EL253 - Sistemas Digitales

Semestre 2020-2

Profesor Kalun José Lau Gan

Sesión de Teoría Semana 7

1

Agenda

- Resolución PC1
- Memorias en VHDL

2

Preguntas por parte de los alumnos

3

Parte 1 de PC1:

- a) **Xilinx**, Intel (ex. **Altera**), Lattice Semiconductor, Microchip (ex. Atmel), Cypress Semiconductor
- b) FPGA mas denso (mas grande): Intel Stratix 10GX 10M
- c) Velocidad y grado (comercial, industrial, militar)
- d) Características del Xilinx Zynq: Dispositivo híbrido (FPGA+Microprocesador) y se le conoce como SoC FPGA. Posee núcleo ARM Cortex-A9. Bajo consumo energético.
- e) IEEE1164 posee nueve estados o niveles lógicos: U, X, 0, 1, Z, W, L, H, -
- f) `signal PC1_REG: std_logic_vector(0 upto 5);`
- g) `SALE <= "110011" when ENT = "000" else
"001100" when ENT = "010" else
"101010" when ENT = "100" else
"ZZZZZZ";`

4

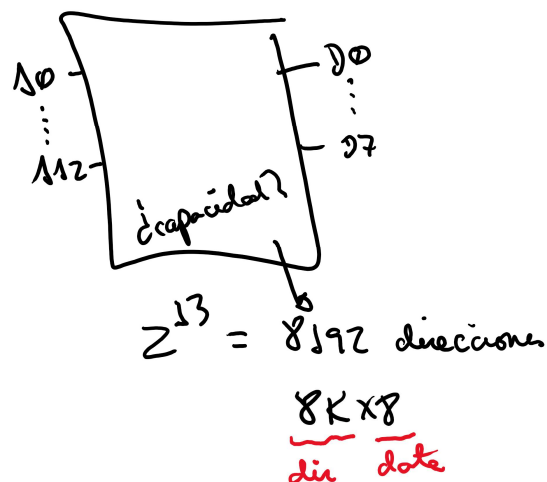
Memorias

- Información se encuentra en el set de slides de la semana 6
- Video: MegaProcessor
<https://www.youtube.com/watch?v=INa9bQRPMB8>
- Video: Manchester Baby
<https://www.youtube.com/watch?v=cozcXiSSkwE>

5

Ejercicios sobre dimensionamiento de memorias

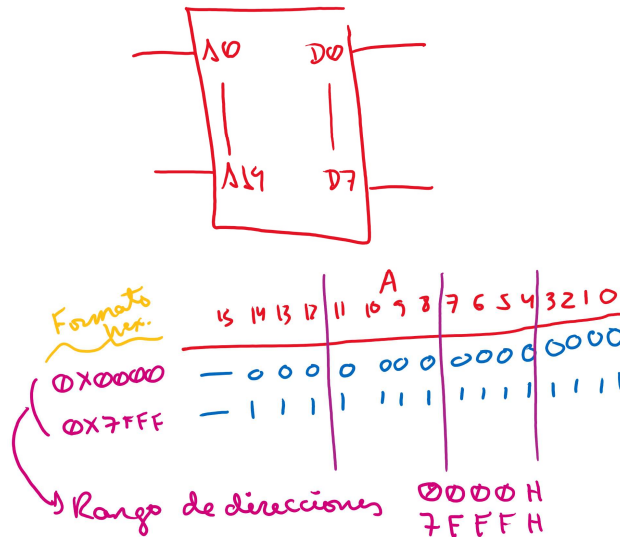
- Qué capacidad tendrá una memoria que posee 13 líneas de direcciones y 8 bits de ancho de datos?



6

Ejercicios sobre dimensionamiento de memorias

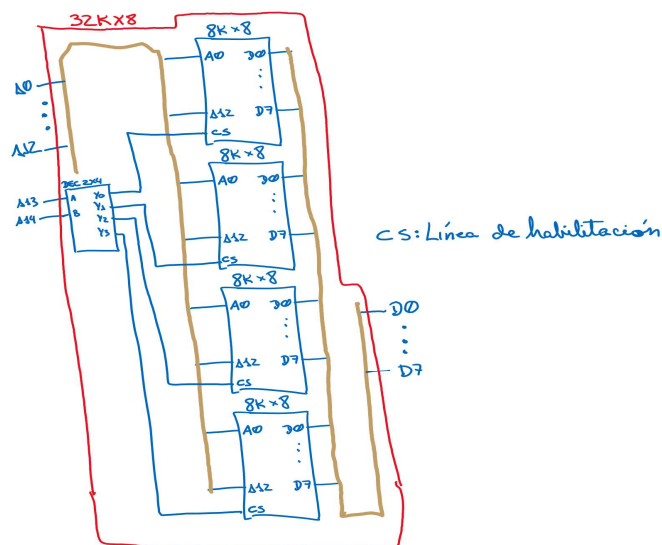
- ¿Cuál es el rango de direcciones de la siguiente memoria?



7

Ejercicios sobre dimensionamiento de memorias

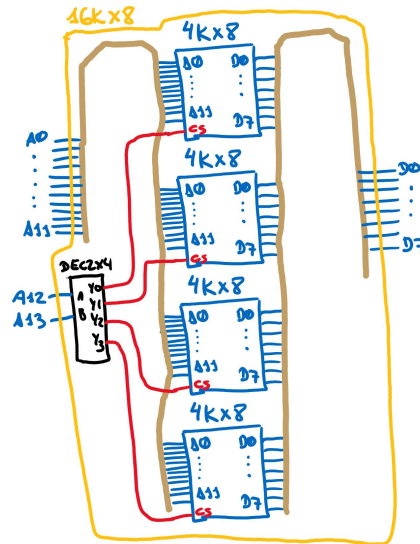
- ¿Cómo construyo una memoria de 32Kx8 con módulos de memoria de 8Kx8?



8

Ejercicios sobre dimensionamiento de memorias

- ¿Cómo construyo una memoria de 16Kx8 con módulos de memoria de 4Kx8?



9

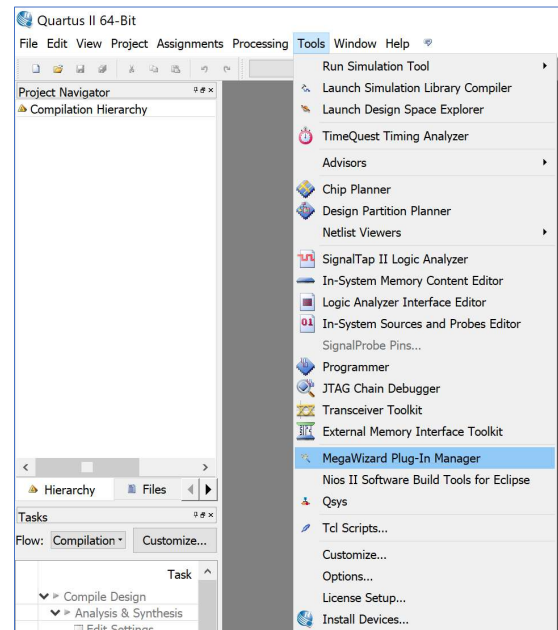
Documentación adicional:

- Memoria RAM 6264 (8Kx8):
<http://users.ece.utexas.edu/~valvano/Datasheets/MCM6264.pdf>
- Memoria EEPROM AT28C64 (8Kx8):
<http://ww1.microchip.com/downloads/en/devicedoc/doc0001h.pdf>

10

Uso de MegaWizard en Altera Quartus II

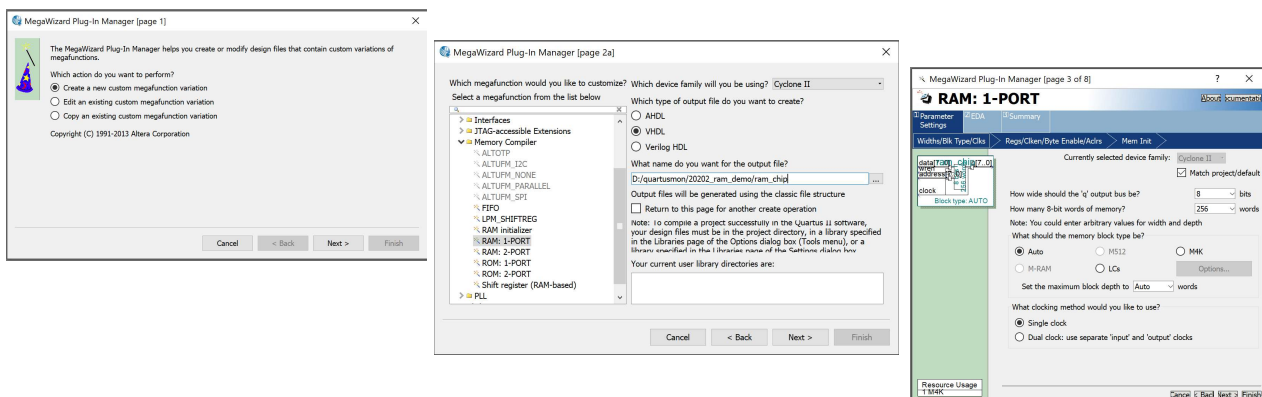
- Aplicación para configurar e integrar módulos digitales prefabricados por Altera en un proyecto.
- Usaremos el MegaWizard para invocar a un módulo de memoria.



11

Instanciación de una RAM: 1-PORT

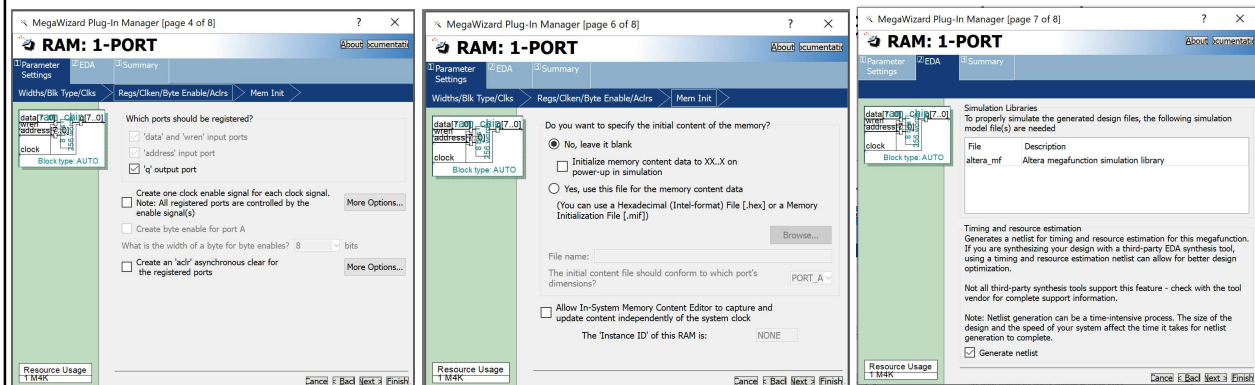
- Antes de usar el MegaWizard se debe de crear un proyecto
- El módulo “RAM: 1-PORT” a crear se alojará como un IP-Core dentro del proyecto creado y se lo instanciará como componente en un archivo VHDL



12

Instanciación de una RAM: 1-PORT

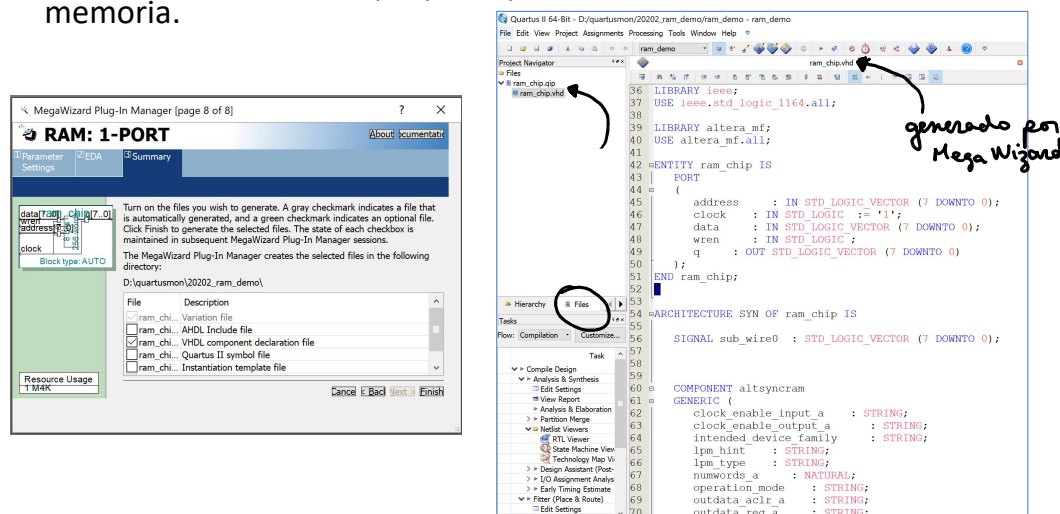
- En este primer ejemplo dejaremos todas las opciones por defecto



13

Instanciación de una RAM: 1-PORT

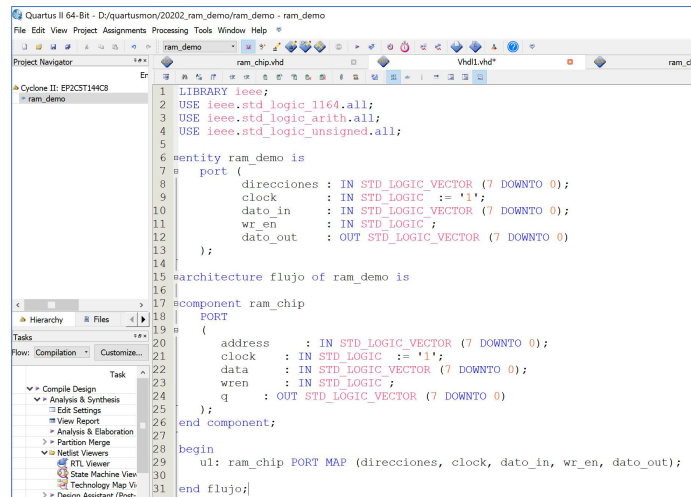
- Al finalizar el MegaWizard revisar los archivos generados, se ubicarán en la pestaña de archivos del proyecto ya que aún no se ha creado el vhd del proyecto para la instanciación de la memoria.



14

Instanciación de una RAM: 1-PORT

- Se creará el vhd principal del proyecto donde se instanciará la memoria creada por el MegaWizard en forma de componente.
- A partir de aquí se podrán conectar diferentes módulos de acuerdo a los requerimientos de la aplicación.



```

1  LIBRARY ieee;
2  USE ieee.std_logic_1164.all;
3  USE ieee.std_logic_arith.all;
4  USE ieee.std_logic_unsigned.all;
5
6  entity ram_demo is
7      port (
8          direcciones : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
9          clock       : IN STD_LOGIC := '1';
10         dato_in      : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
11         wr_en        : IN STD_LOGIC;
12         dato_out     : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
13     );
14
15 architecture flujo of ram_demo is
16
17     component ram_chip
18     PORT
19     (
20         address : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
21         clock    : IN STD_LOGIC := '1';
22         data     : IN STD_LOGIC_VECTOR (7 DOWNTO 0);
23         wren     : IN STD_LOGIC;
24         q       : OUT STD_LOGIC_VECTOR (7 DOWNTO 0)
25     );
26 end component;
27
28 begin
29     ul: ram_chip PORT MAP (direcciones, clock, dato_in, wr_en, dato_out);
30
31 end flujo;
  
```

15

Fin de la sesión de teoría

16