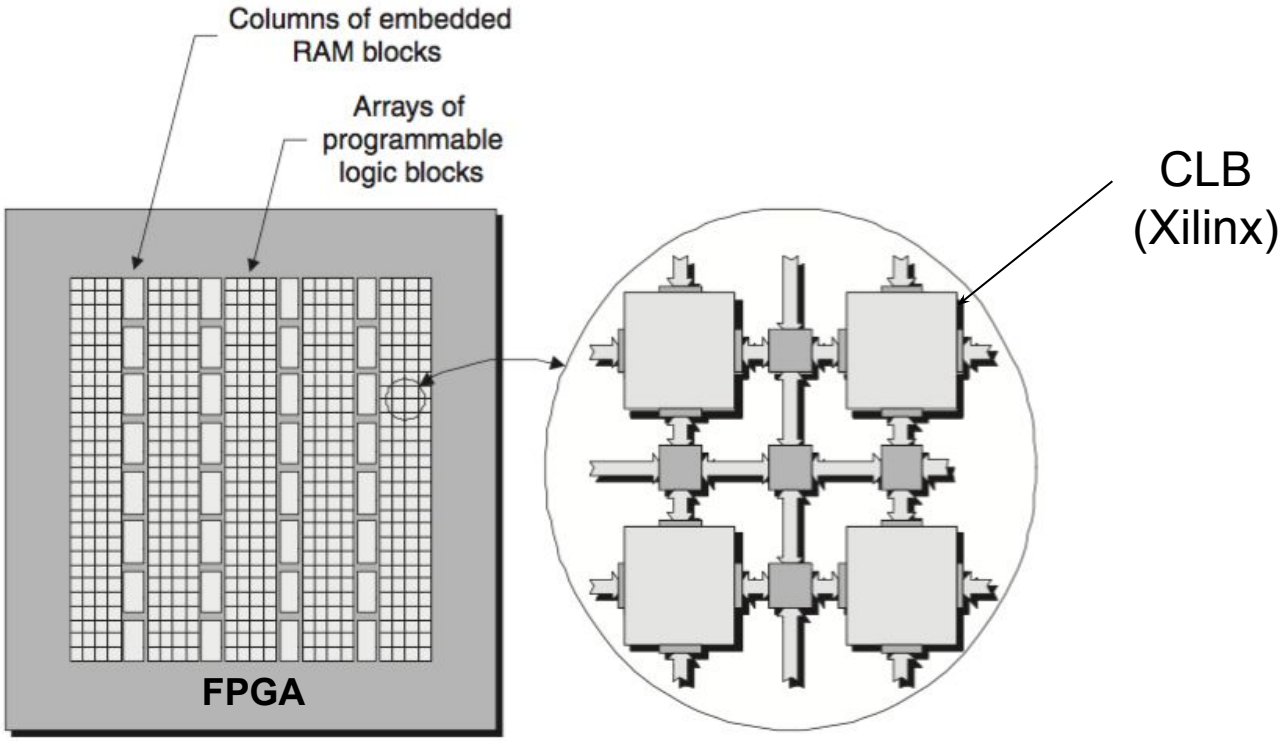
Trabalho 1 – parte 2

- A tarefa deste trabalho será fazer a prototipação em FPGA do processador MIPS
  - Utilizaremos as Block RAMs do FPGA para implementar a memórias de instruções e dados



Trabalho 1 – parte 2

- A memória fornecida (Memory.vhd) está descrita de maneira que o sintetizador (XST) faça a inferência de *block* RAMs
- Desta maneira as memórias serão implementadas utilizando os blocos de RAM internos ao FPGA ao invés de utilizar LUTs
- O número de palavras das memórias é especificados através do parâmetro generic SIZE

```
DATA_MEMORY: entity work.Memory
        generic map (
            SIZE => 128,     -- Memory depth
            ...
```

Disponibilidade de Block RAM no FPGA Spartan-6 LX16: 72000 Bytes

Trabalho 1 – parte 2

- O arquivo de imagem pode ser gerado pelo MARS (mesmo arquivo utilizado na simulação VHDL)
- Cada linha do arquivo de imagem da memória corresponde a uma palavra
- O arquivo de imagem das memórias deve conter um número de linhas igual ou maior que a profundidade da memória (parâmetro SIZE)
- Exemplo

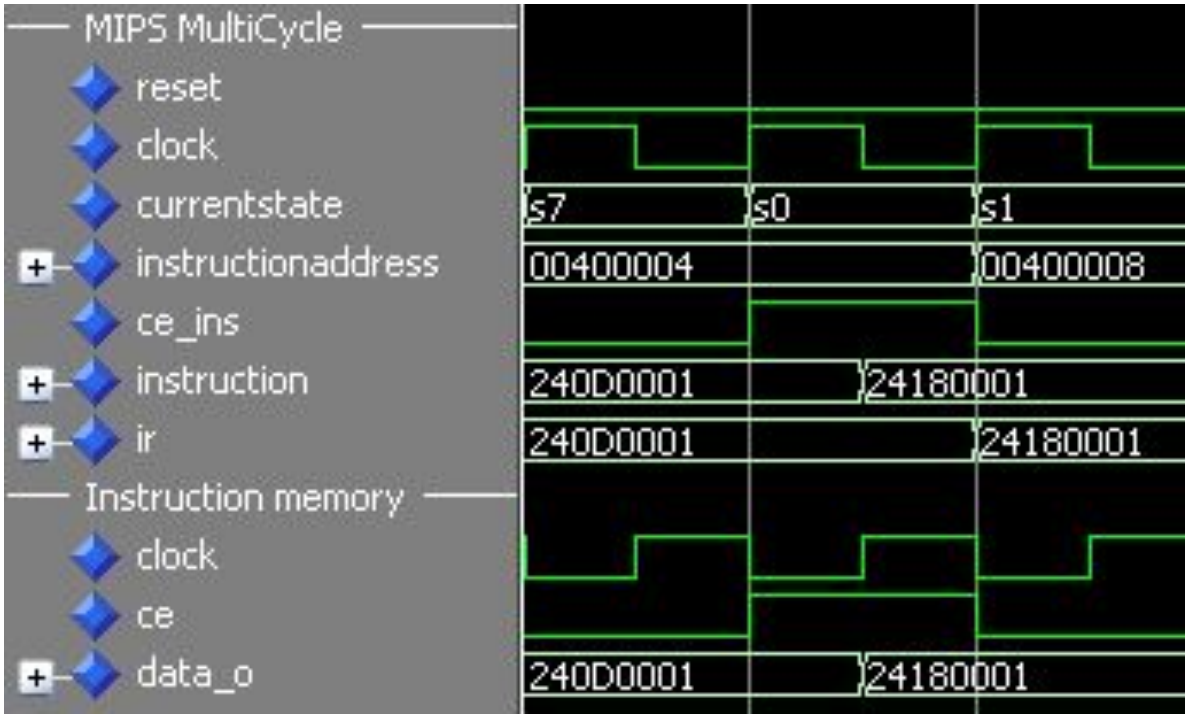
```
DATA_MEMORY: entity work.Memory
        generic map (
            SIZE => 128,     -- Memory depth
            ...
```

O arquivo de imagem deve ter pelo menos 128 linhas

Trabalho 1 – parte 2

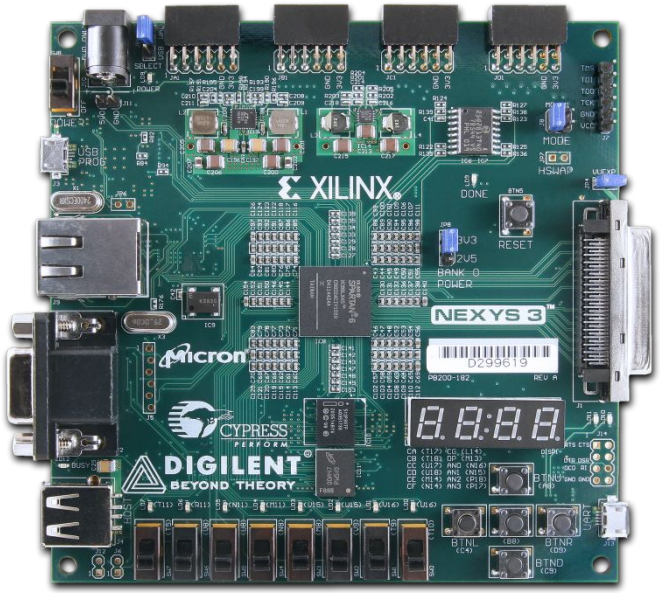
- Visto que as *block* RAMs do FPGA tem leitura síncrona, é necessário que ela opere nas bordas de descida do *clock* do processador para fins de sincronia

Ajustar comentários relativos à leitura em Memory.vhd a fim de tornar a leitura síncrona



Trabalho 1 – parte 2

- A prototipação será feita utilizando a placa Nexys 3 (NUPEDEE)
  - Nesta placa o FPGA opera a 100MHz, no entanto nosso projeto deve operar a 50 MHz
  - Para dividir a frequência da placa utilizaremos um dos DCMs (*Digital Clock Manager*) do FPGA



Trabalho 1 – parte 2

- Para utilizar o DCM do FPGA, deve-se adicionar ao projeto componente *ClockManager*
  - ClockManager.vhd (moodle)

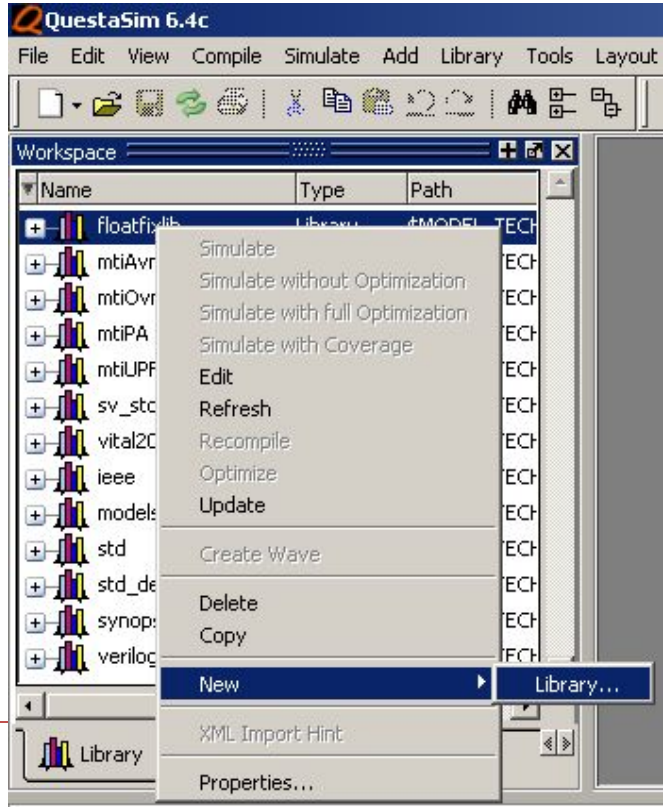
```
entity ClockManager is
    port (
        -- Board clock (100MHz)
        clk_in              : in     std_logic;

        -- Project clock
        clk_div2            : out    std_logic;
        clk_div4            : out    std_logic
     );
end ClockManager;
```

clk_in / 2 (processador)

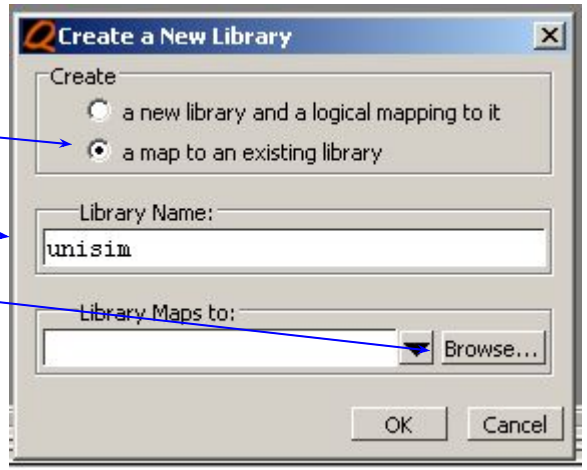
Trabalho 1 – parte 2

- Para simular componentes internos do FPGA (e.g. DCM) será necessário utilizar a biblioteca UNISIM
  - Moodle: unisim.zip (descompactar)
- Para adicionar biblioteca no Modelsim
  - Abrir o menu de contexto do painel *Library* e selecionar *New → Library*



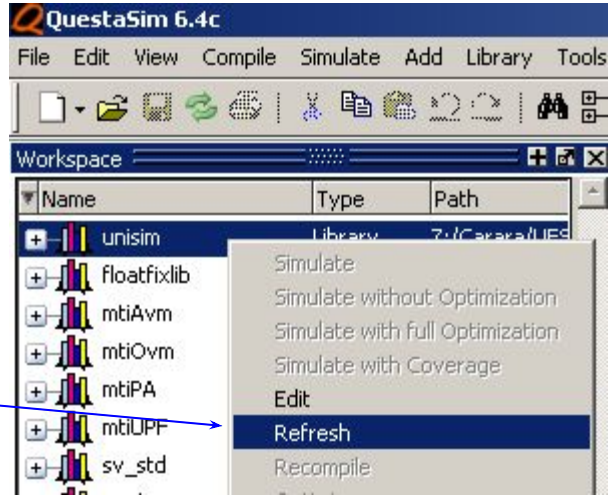
Trabalho 1 – parte 2

- Para simular componentes internos do FPGA (e.g. DCM) será necessário utilizar a biblioteca UNISIM
  - Moodle: unisim.zip (descompactar)
- Para adicionar biblioteca no Modelsim
  - Abrir o menu de contexto do painel *Library* e selecionar *New → Library*
  - Marcar a opção *"a map to an existing library"*, Adicionar o nome da *library* Selecionar o diretório da *library*



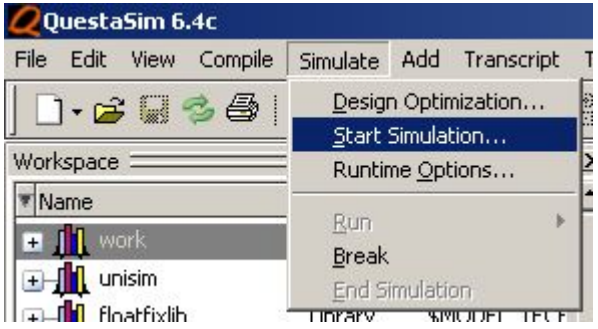
Trabalho 1 – parte 2

- Para simular componentes internos do FPGA (e.g. DCM) será necessário utilizar a biblioteca UNISIM
  - Moodle: unisim.zip (descompactar)
- Para adicionar biblioteca no Modelsim
  - Abrir o menu de contexto do painel *Library* e selecionar *New → Library*
  - Marcar a opção *"a map to an existing library"*, Adicionar o nome da *library* Selecionar o diretório da *library*
  - Compilar (*Refresh*)
    - É necessesário compilar apenas uma vez



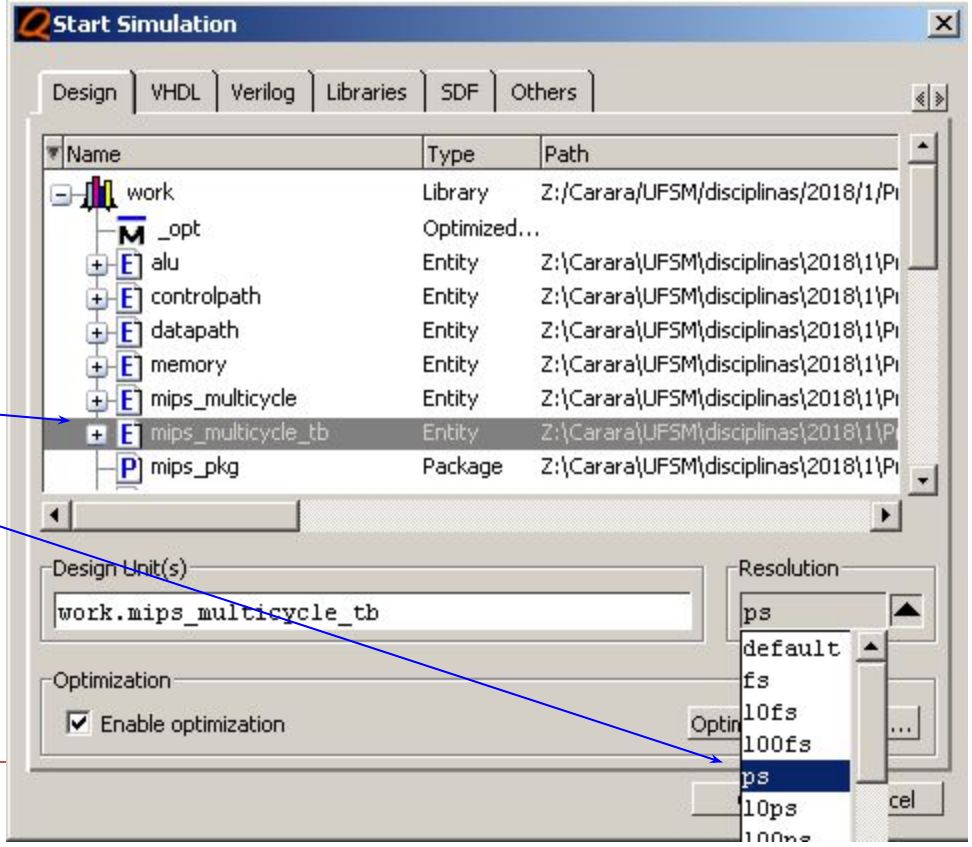
Trabalho 1 – parte 2

- Para simular projetos que utilizam o DCM, é necessário alterar a resolução do simulador Modelsim antes de iniciar a simulação



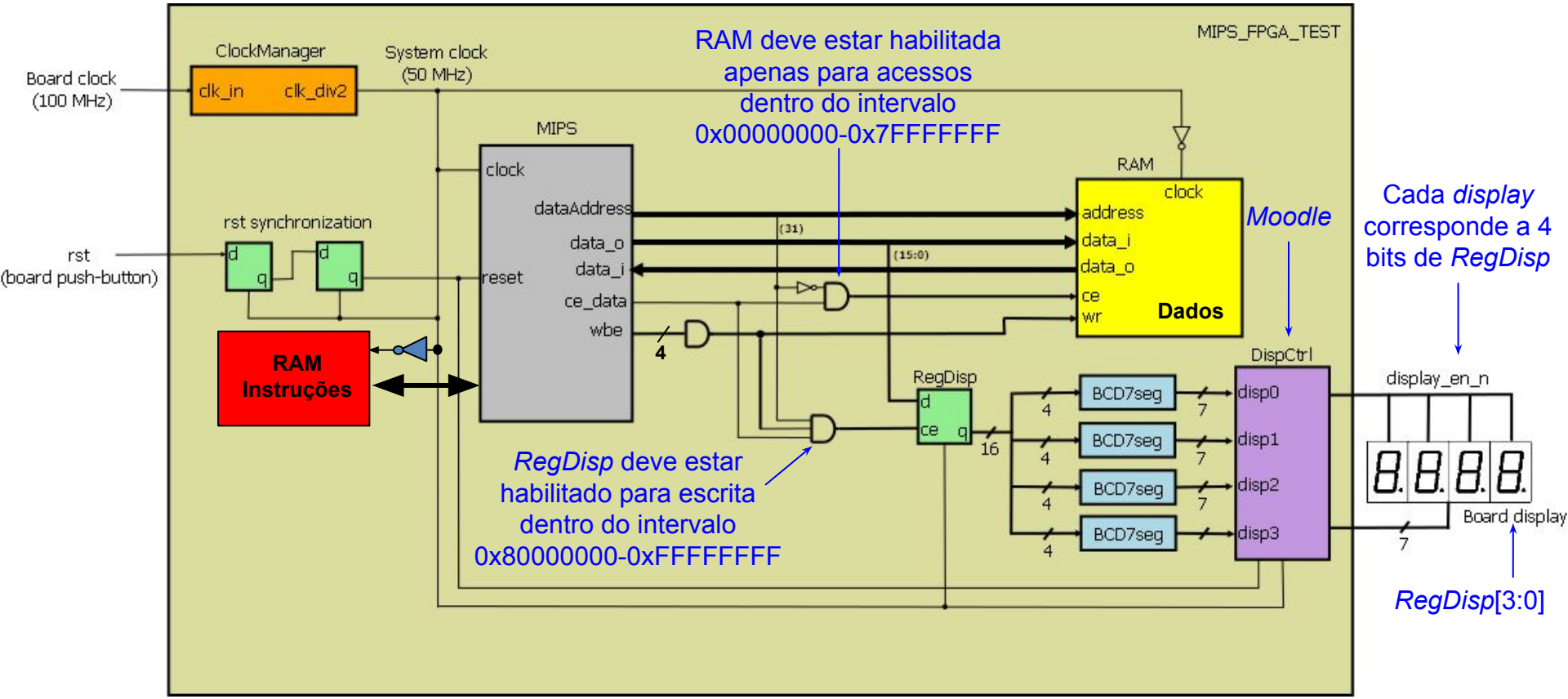
Selecionar o *test bench*

Setar a resolução para *ps*



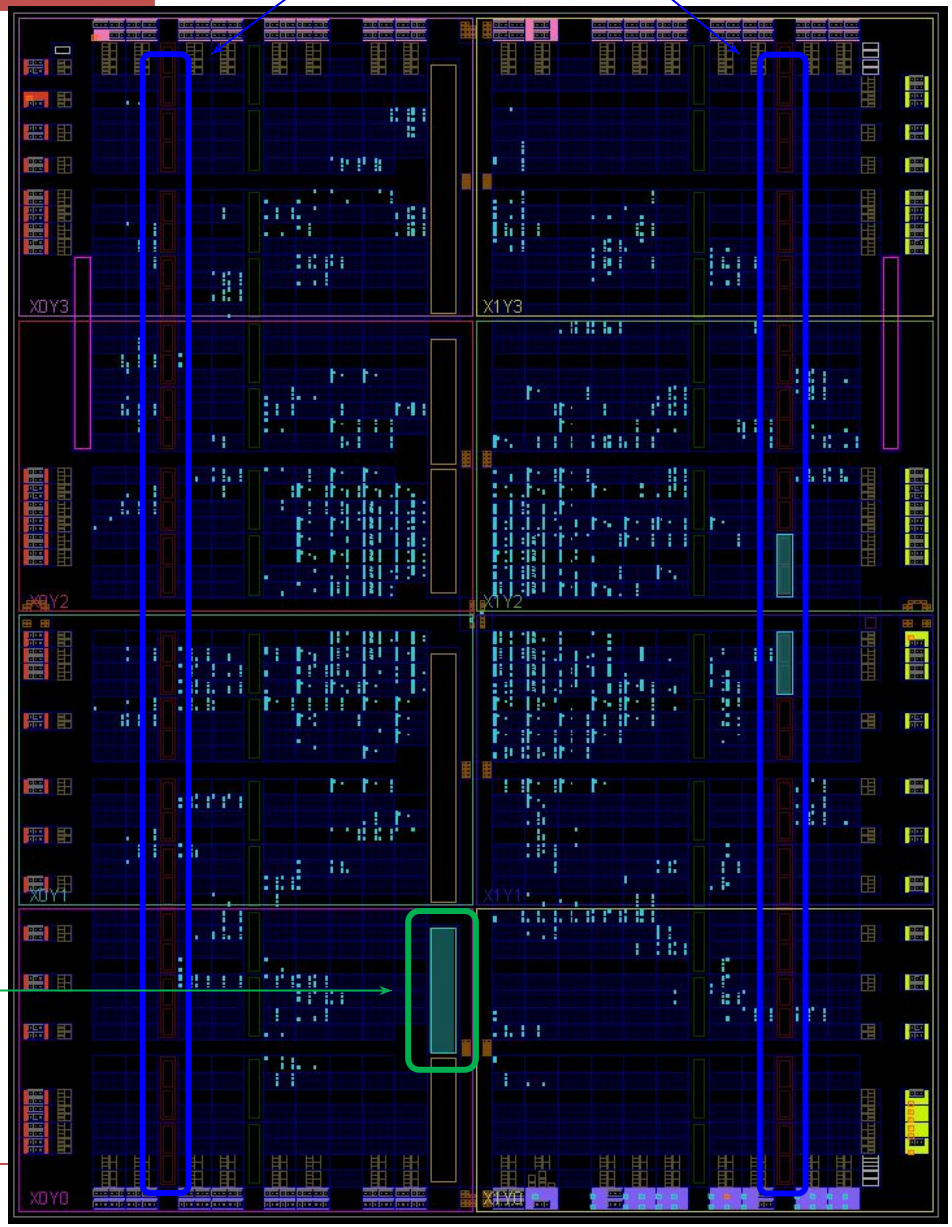
Trabalho 1 – parte 2

- Para que seja possível verificar o funcionamento da prototipação utilizaremos os *displays* da placa



Trabalho 1 – parte 2

- *Implemented design*
  - Spartan 6 (LX16)



Trabalho 1 – parte 2

- Aplicação
  - Implementar um contador hexadecimal com incremento de 1 segundo
  - Considerar o tempo de execução das instruções e a frequência a fim de atingir um incremento o mais próximo possível de 1 segundo
- Sequência sugerida para o desenvolvimento
  1. Adicionar DCM e sincronização do *reset* ao projeto atual e simular
  2. Criar a *entity* MIPS_FPGA_TEST contendo MIPS, memórias, DCM e sincronização do *reset*. Criar *test bench* para gerar *clock* e *reset* para a *entity* MIPS_FPGA_TEST e simular
  3. Adicionar a parte relativa à interface com os *displays* e simular
  4. Prototipar
     - Os arquivos de imagem das memórias devem estar no mesmo diretório do arquivo Memory.vhd
     - Atenção aos *warnigs!*