µProcessador 8 Programação da FPGA

Mesmo depois do capítulo final, vem o *remake*.

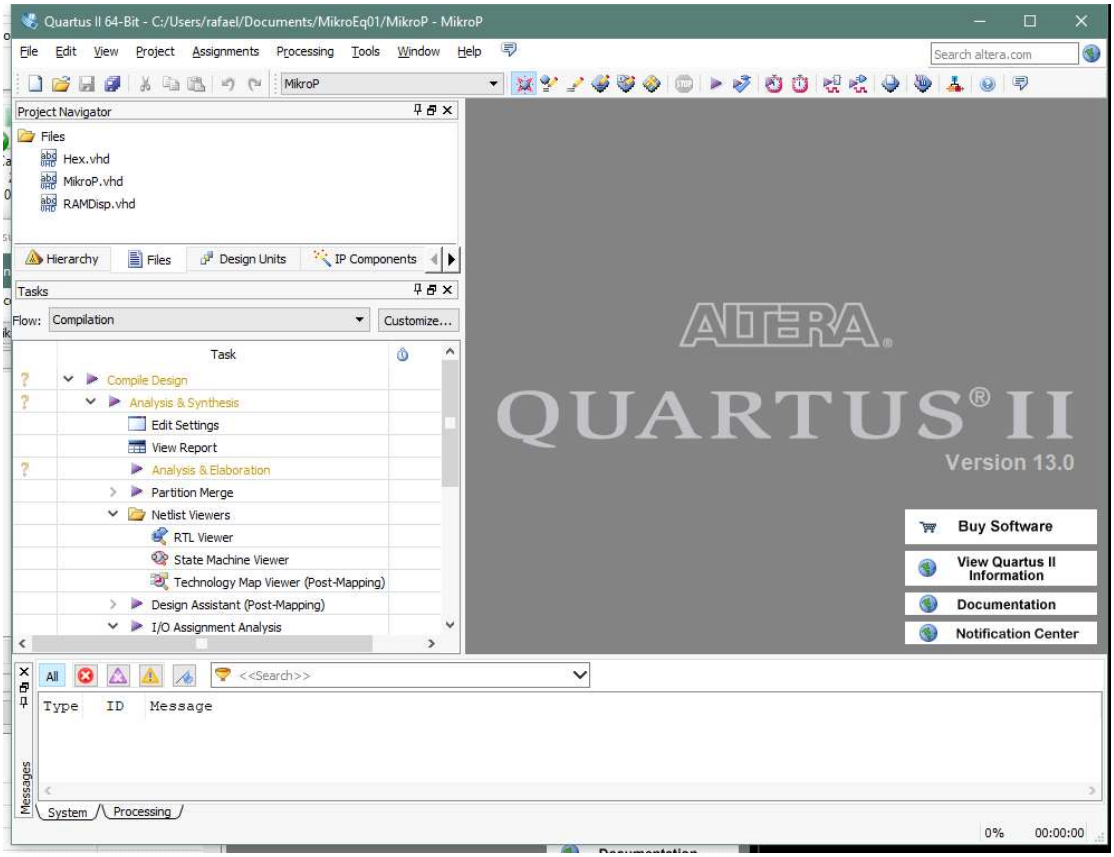
Com tudo simulado e funcionando como esperado, vamos sair da tela do gtkwave para o mundo real! Para permitir que o projeto desenvolvido no GHDL seja gravado numa FPGA (Cyclone II da Altera – EP2C20F484C7) será usado o ambiente integrado Quartus. Vamos usar a placa CycloneII FPGA Starter Board. Para este componente deverá ser usada a versão 13, service pack1 (o Quartus Prime só funciona para a Cyclone IV em diante).

Para acelerar a implementação o ponto de partida um projeto exemplo é disponibilizado. Nele já os pinos físicos já estão mapeados para funções básicas como clock, chaves de reset, chaves de configuração e LEDs de diagnóstico.

De modo similar ao que consideramos até aqui para a geração de estímulos utilizando test bench como *toplevel*, vamos usar os sinais da placa. O esquemático da placa está no diretório do exemplo.

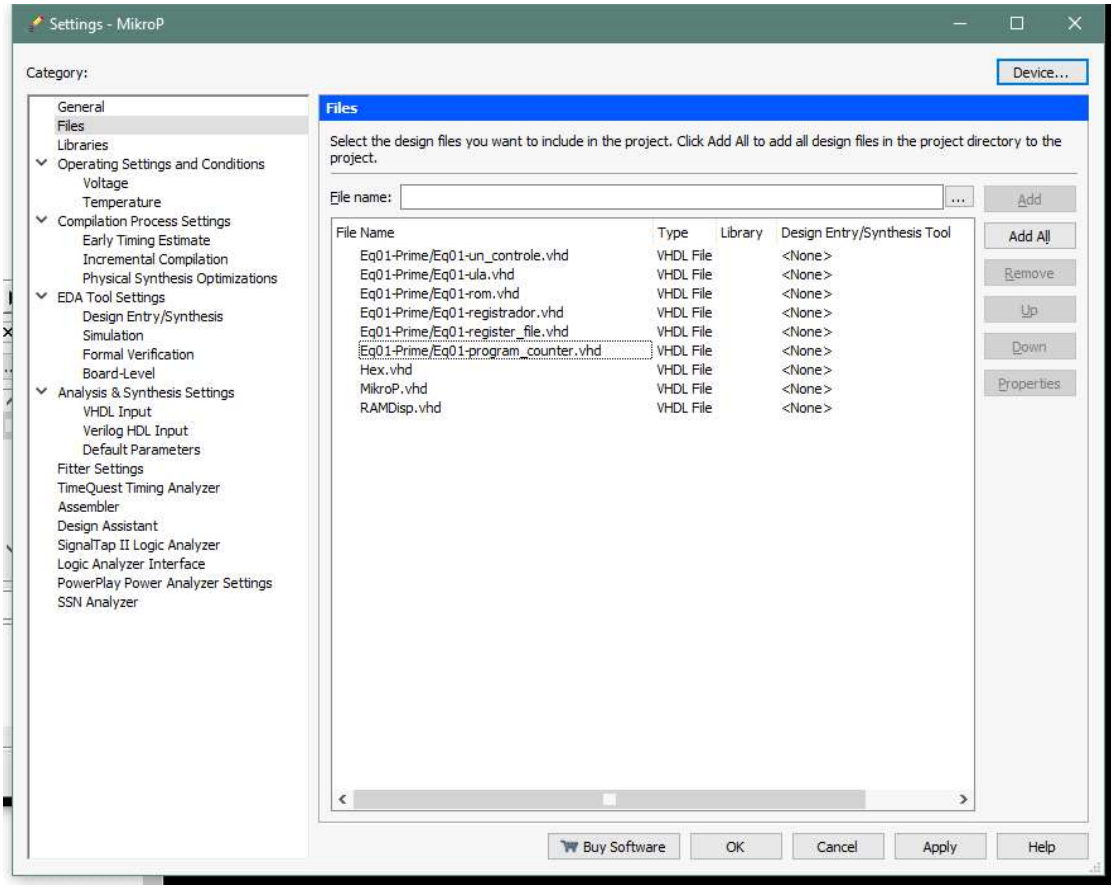
Será necessário, portanto, incluir os arquivos em VHDL do projeto atual (sintetizado pelo *makefile* para GHDL desenvolvido pela equipe) diretamente no Quartus, porém sem o test bench. Além disso, usaremos uma RAM modificada, a qual chamamos de RAMDisp.

Inicialmente, após baixar e descompactar o projeto exemplo na pasta de trabalho, deve-se incluir a pasta inteira do projeto do Lab7 como um subdiretório do mesmo.



Com o projeto exemplo aberto, a partir do menu:
Project > Add/Remove Files in Project…

O diálogo para a inclusão dos arquivos é aberto. Na categoria ***Files*** , clicar no botão "***...***" ao lado do ***File Name:.*** A lista de arquivos a ser incluída é aberta. Apontar para o subdiretório com os arquivos da equipe gerados no Lab7 e selecionar apenas os arquivos .vhd que não sejam test benches. O arquivo que define a RAM fornecido no enunciado do Lab7 não deve ser incluído.



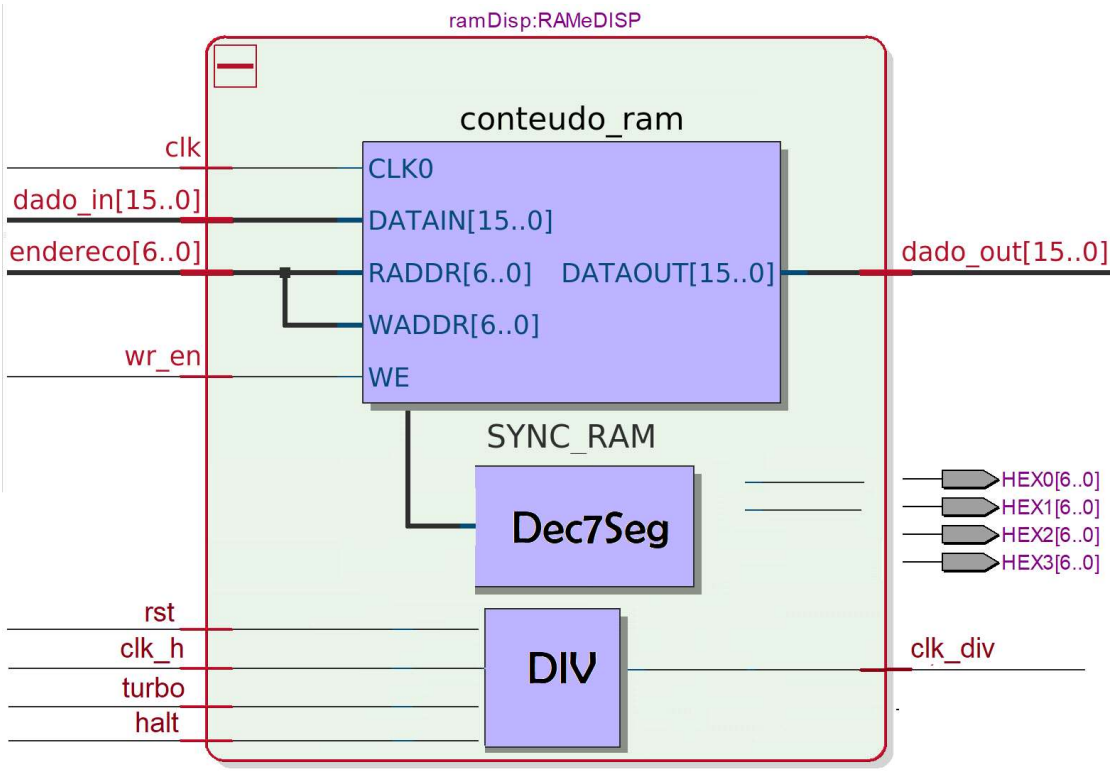
Após incluídos os arquivos da equipe, verificando em detalhes o projeto exemplo, o toplevel mapeia os sinais rst e clk, que eram gerados pelo testbench para a RAM modificada.

Além do display este componente encapsula um divisor de clock, responsável por permitir que possamos rodar o processador a uma velocidade muito baixa. Isso permite que possamos acompanhar a execução do código contido em ROM.

Por conta disso, é necessário acertar o arquivo do procesador para usar os sinais de clock dividido (clk_div). Tudo pode ser feito nos *port maps*.

Além destes, alguns LEDs indicam apenas um contador global de ciclos e chaves de parada (halt) e “turbo”. A história da tecla turbo é bastante interessante no contexto da disciplina. Vale a leitura do verbete:
https://en.wikipedia.org/wiki/Turbo_button

A nova versão da RAM já incluido no projeto (RAMDisp) tem também um periférico mapeado em memória. Neste caso, ao escrever no endereço 127d, o dado é apresentado em decimal num par de displays de 7 segmentos da placa Cyclone II Starter Board. A conversão é feita no modo flash (em um ciclo) para simplificar a temporização e evitar potenciais problemas com a temporização do processador implementado. O display corresponde, portanto, aos rudimentos de uma memória de vídeo. As escritas são feitas em um endereço específico e o mapeamento e varredura ficam encapsulados neste componente.

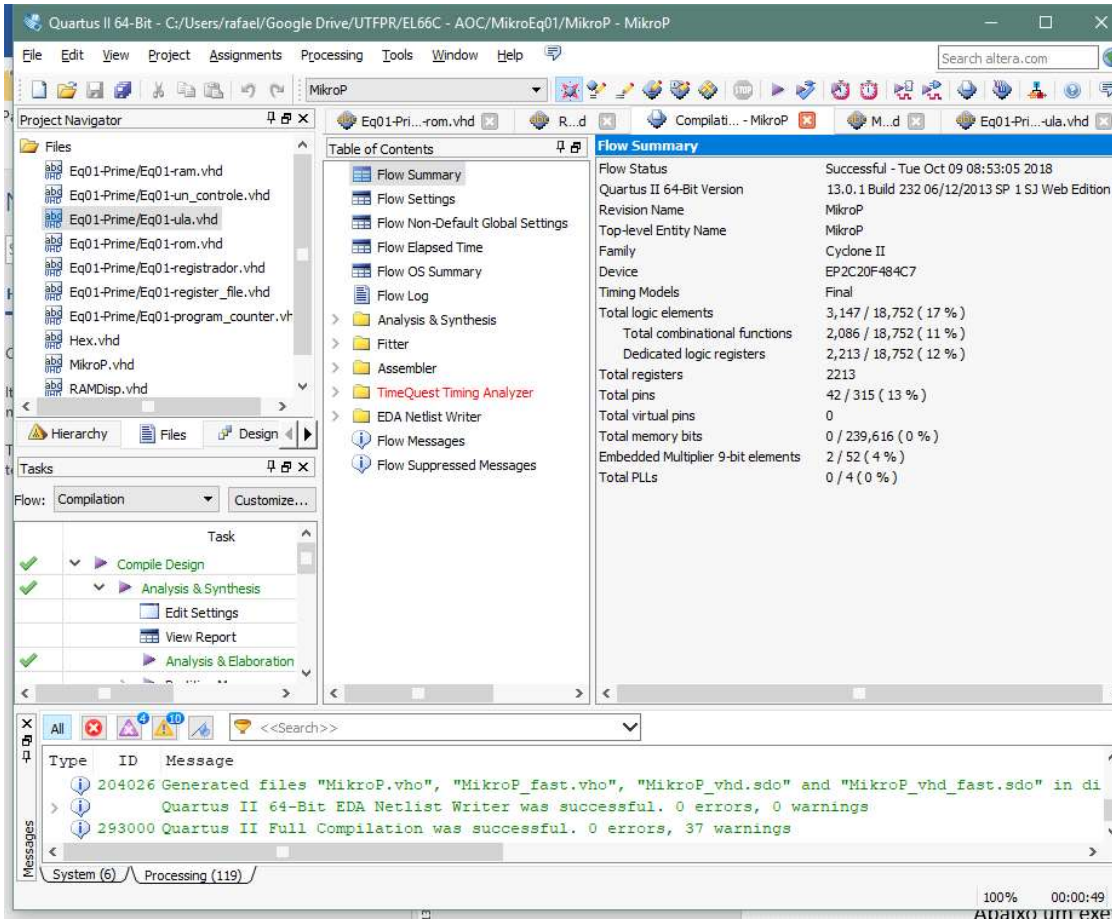


Em VHDL, a nova interface da RAM fica como a seguir.

```
entity ram is
   port(
         clk      : in std_logic;
         endereco : in unsigned(6 downto 0);
         wr_en    : in std_logic;
         dado_in  : in unsigned(15 downto 0);
         dado_out : out unsigned(15 downto 0);

          HEX0      : out std_logic_vector (6 downto 0);
          HEX1      : out std_logic_vector (6 downto 0);
          HEX2      : out std_logic_vector (6 downto 0);
          HEX3      : out std_logic_vector (6 downto 0);
          halt     : in std_logic;
          turbo    : in std_logic;
          clk_h    : in std_logic;
          clk_div  : out std_logic;
          rst      : in std_logic;
   );
end entity;
```

Ajuste os arquivos de seu processador para esta nova RAM e o divisor de clock e “execute a síntese”. Para isso, usar o menu ***Proccessing > Start Compilation***. Observe no relatório final da síntese a quantidade de elementos lógicos utilizados.



Alguns Warnings (novos) podem ocorrer. A maioria diz respeito apenas ao fato de ser usada uma versão de avaliação do pacote de software de síntese Quartus e detalhes de simulação que não foram definidos.

Se todos os sinais foram ligados corretamente, a síntese do circuito gerará um arquivo que pode ser gravado no componente. Ter certeza disso pode levar algum tempo e a necessidade de ir colocando sinais de diagnóstico em LEDs caso o projeto não esteja bem organizado.

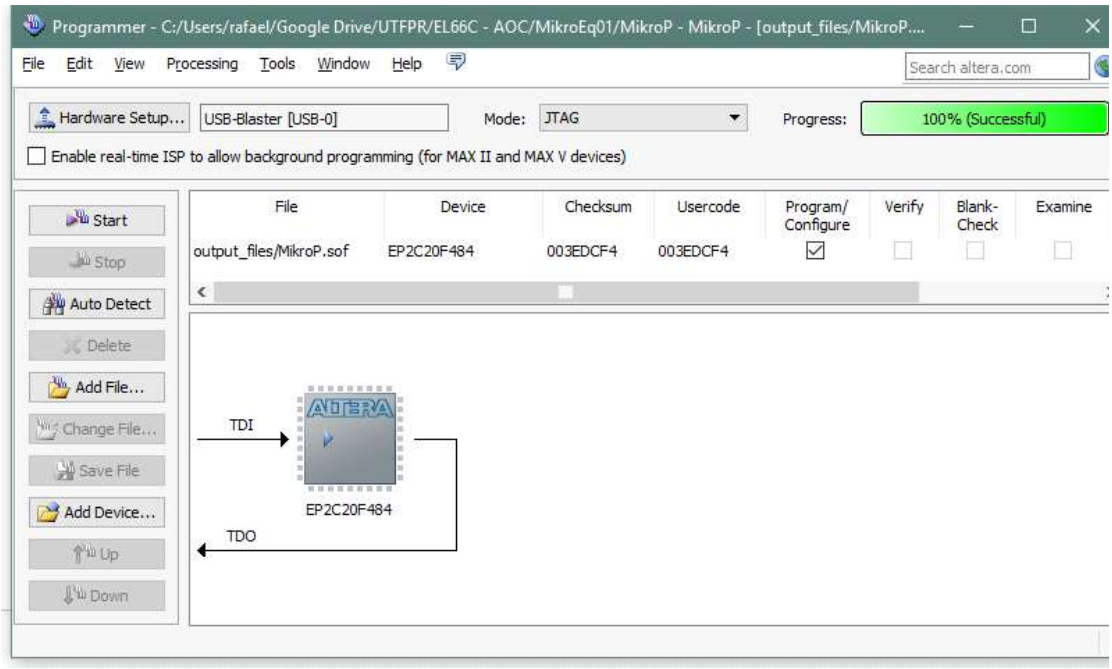
Dica: Mapear o decoder do display para o contador de programa (PC) pode ser uma boa opção para saber se o programa da ROM está executando. Recursos mais avançados como ler chaves para implementar esperas no programa seriam interessantes. O caminho que isso leva é o desenvolvimento de uma espécie de BIOS para a placa. Porém, esse não é o foco deste laboratório de encerramento da disciplina.

Para gravar o arquivo gerado: MikroP.sof basta usar o programador (menu

Tools > Programmer). Abaixo é mostrada a tela do Programador.

Clique duas vezes no arquivo .sof para ter certeza de que a versão no diretório corrente de trabalho está sendo usada. Com a placa conectada a qualquer porta USB, use o botão "***Auto Detect***" e o componente deve aparecer conforme a figura, com um trângulo azul sobre o chip.

A chave SW12 da placa deve estar na posição "RUN". Clique no botão ***Start*** para descarregar o arquivo para o componente.



Após a gravação o processador roda diretamente o programa definido na ROM. Lembrando que o reset está mapeado no botão KEY0 e as chaves de halt e turbo estão mapeadas nas chaves SW9 e SW8, respectivamente. Se tudo correu bem até aqui, sorria! O resultado da execução do algoritmo de números primos é mostrado no display do mundo real.

Apesar das chaves de halt e turbo poderem auxiliar, outros recursos tais como breakpoints de HW e portas de trace podem ser implementados. Mas, deste ponto em diante a desenvolvimento, em geral, passa pelo uso mais intenso de IP (Intelectual Property), arquivos com estruturas de circuitos que podem ser licenciados. Nestes casos, torna-se relevante a aderência a bibliotecas e conjunto de ferramentas específicas de fabricantes de FPGA (Altera, Xilinx, Lattice, Anchronix, Microchip, etc.). Uma referência interessante do ponto de vista de IP, é a Propriedade Intelectual aberta (https://opencores.org). À parte da contradição na tradução literal, trata-se de um movimento muito promissor, diga-se de passagem.

Arquivos a Entregar

Um único ZIP contendo toda a pasta do projeto em Quartus incluindo o

processador da equipe para que o arquivo fique menor, pode apagar a pasta “*simulation*”.