Laboratório 12

1. Quando for demonstrar seu trabalho, tome nota do número da placa utilizada. O número da placa será utilizado para atribuir a nota ao grupo.

2. A última página deste documento contém um checklist com todos os arquivos que fazem parte da entrega.

3. Os nomes dos arquivos devem ser seguidos, e isso faz parte da avaliação.

4. A entrega deverá estar em único arquivo .ZIP, com o nome **T_Lab12_RA.zip**, **T** é a turma, e **RA** é o RA do componente do grupo que fará a entrega. Por exemplo, B_Lab12_123456.zip é a entrega do grupo do aluno com o RA 123456, na turma B.

5. Não divida ou agrupe em pastas os arquivos dentro do .ZIP.

6. A entrega deve ser feita pelo Google Forms (https://forms.gle/qBnoCDXBQec8tpvE6). Você deve estar autenticado com uma conta do Google - pode ser uma conta pessoal ou da DAC.

7. Apenas um integrante do grupo precisa fazer a entrega.

8. Preste especial atenção aos nomes das entidades e sinais (entradas e saídas) descritos nos laboratórios. Isso também faz parte da avaliação.

9. Se mais do que um arquivo for recebido para a mesma entrega, o último recebido será considerado. Utilize o mesmo RA do aluno entregando.

10. A última página deste documento contém um checklist com todos os arquivos que fazem parte da entrega.

11. Faça o download do arquivo **lab12_m1ps_v2020.1.qar**, que contém um projeto arquivado do Quartus em que este laboratório se baseia.

Parte I - Conceitos iniciais

Este laboratório visa exercitar os conhecimentos sobre processadores. A plataforma utilizada é o processador m1ps (meu 1º processador simples), conforme o diagrama de blocos da Figura 1, e o hardware de monitoramento mM (m1ps Monitor), conforme descrito em aula.

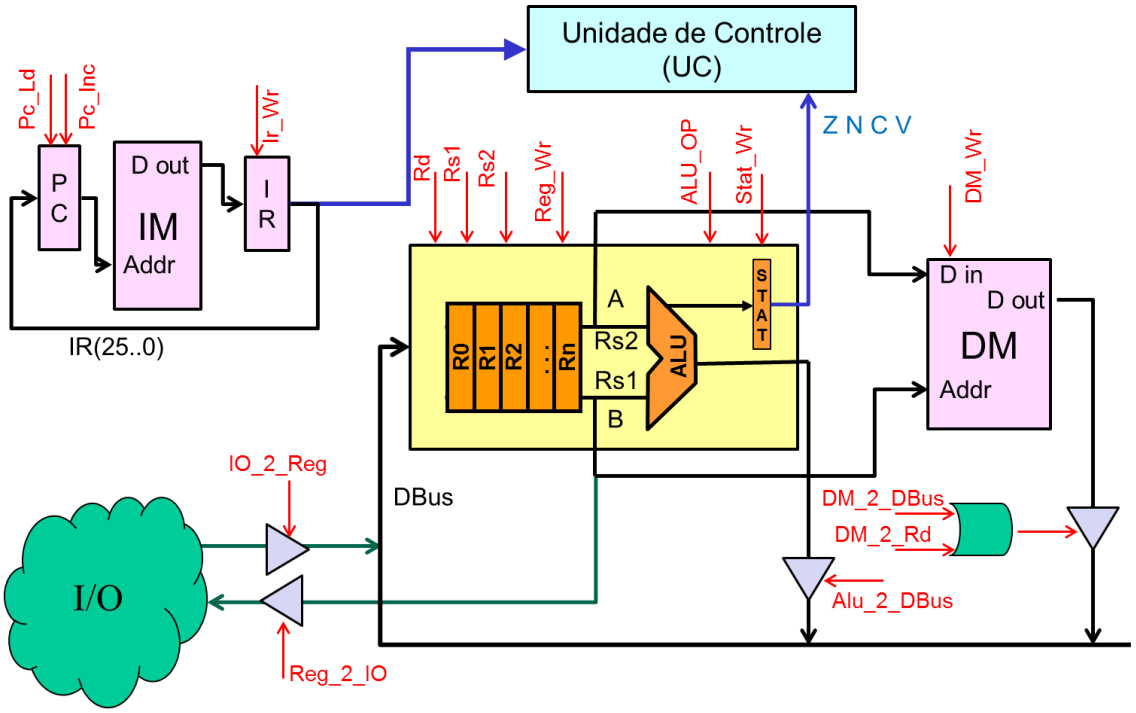

Figura 1 - Diagrama de Blocos do m1ps

Faça o download do arquivo *lab14_m1ps_v2019.1.qar*. Ele é um arquivo de projeto do Quartus arquivado e contém o código-fonte - incompleto - do processador, o código do componente *m1ps monitor*, programas que computam a sequência de Fibonacci e arquivos auxiliares de simulação, configuração e acompanhamento.

Após restaurar o projeto no Quartus (*Project > Restore Archived Project*), o diretório alvo terá a seguinte estrutura:

- **mM**: diretório contendo os códigos VHDL do componente *m1ps monitor*;
- **Processor**: diretório contendo o código-fonte do processador;
  - *fibonacci*.mif*: imagens de memória com os programas compilados;
- **CPU_sim.vwf**: arquivo de entrada para simulação do processador;
- **fibonacci*.m1ps**: fontes Assembly dos programas;

- **fibonacci*.xls**: planilhas para acompanhamendo de simulação e execução, contendo cada instrução executada, sua representação em binário e hexadecimal, e os números de Fibonacci esperados.
- **m1ps.qpf**: arquivo de projeto do Quartus;
- **mM_toplevel.qsf**: configurações do projeto - já contém os assignments de pinos para gravação na placa.

I.1. Arquivos BDF:

Arquivos BDF (Block Diagram File) são arquivos de diagrama de blocos do Quartus. Com eles, as entidades são descritas de maneira gráfica e em alto nível, “desenhando” o componente desejado na tela.

Neste laboratório, o arquivo que representa a entidade principal do processador m1ps é um arquivo BDF, o **CPU.bdf**, no diretório **Processor**. Para concluir o laboratório, você precisará fazer edições nesse arquivo.

A primeira modificação fundamental é a escolha do programa que o processador executará. Isso é feito mudando o parâmetro de arquivo de imagem de memória no componente I-Memory do processador, em cada tarefa de execução (partes **IV**, **VII** e **IX**). Para isso, abra o arquivo **CPU.bdf**, procure o componente I-Memory e clique duas vezes sobre a tabela de parâmetros acima dele. Na caixa de diálogo que surge, selecione a aba *Parameter* e altere o valor do parâmetro *MIF_FILE* conforme necessário.

Além dessa modificação, você poderá sentir a necessidade de alterar o fluxo de dados do processador para cumprir as tarefas de incluir novas instruções no processador (partes **VI** e **VIII**). Para isso, basta utilizar as ferramentas da barra de tarefas para “desenhar” fios, adicionar portas lógicas, etc.

I.2. Simulações com arquivo VWF:

Arquivos VWF são ferramentas do Quartus para simulação de componentes com ondas pré-definidas de entrada que utilizam o Modelsim como *backend*, abstraindo a necessidade de *testbenches* e scripts de execução. Neste laboratório, foi preparada uma onda de simulação que descreve, para o processador m1ps, as operações de entrada e saída necessárias para correta execução dos códigos para geração das sequências de Fibonacci. Na forma de onda da simulação, poderão ser vistos os barramentos de entrada e saída do processador, onde é escrita a sequência de Fibonacci, e vários sinais de controle.

Para proceder com a simulação, a entidade *top-level* do projeto deve ser o arquivo **CPU.bdf** e a tarefa de compilação **Analysis & Synthesis** deve ter sido executada. Daí, basta abrir o arquivo **CPU_sim.vwf** (clicando duas vezes) e disparar a simulação em *Simulation* > *Run Functional Simulation*. Como o projeto do processador é relativamente grande, essa simulação pode levar um pouco mais de tempo que o normal.

Parte II - Banco de registradores

Crie um banco de registradores para o m1ps, tomando como base o projeto feito no Laboratório 07. Modifique-o (se necessário) para que ele siga as seguintes especificações:

1) Permita duas leituras simultâneas;
2) O registrador r0 não pode ter seu valor alterado, nunca;
3) Toda leitura endereçada ao registrador r0 retorna 0;
4) Os registradores têm escrita síncrona (na borda de subida do clock) e leitura assíncrona (independente do clock);
5) O sinal de clear está ativo em nível lógico alto;
6) Quando o sinal de leitura (*RD_EN*) estiver desativado (nível lógico baixo), as portas de saída (leitura) do banco devem estar em alta impedância;
7) Note os parâmetros de configuração definidos na seção *Generic* da declaração da entidade.

A entidade principal de seu banco de registradores deve ser adicionada ao projeto no arquivo **bank.vhd**, no diretório **Processor**. O arquivo já contém, descrita, a interface de comunicação com o processador.

Parte III - Memória

Baseando-se no projeto da memória feito no Laboratório 13, modifique-o (se necessário) para que ele siga as especificações da Memória de Instrução (IM) e da Memória de Dados (DM) do processador m1ps. A entidade principal de sua memória deve ser adicionada ao projeto no arquivo **memory.vhd** no diretório **Processor.** O arquivo já contém, descrita, a interface de comunicação com o processador, e o código necessário para inicialização da memória a partir de um arquivo.

Especificações adicionais da memória:

1) Uma porta de dados de escrita e uma porta de dados de leitura independentes;
2) Escrita síncrona quando o sinal de permissão de escrita (*we*) está em nível lógico alto;
3) Leitura síncrona a cada ciclo de clock (sem sinal de permissão de leitura);
4) Inicializável com um arquivo *.mif*.
5) Note os parâmetros de configuração definidos na seção *Generic* da declaração da entidade.

Parte IV - Primeira execução do código de Fibonacci

Altere a entidade top-level do projeto para a entidade **CPU** (Processor/CPU.bdf) para simular o comportamento do processador. Utilize o arquivo **CPU_sim.vwf** para simulação e certifique-se de que as instruções estão sendo executadas corretamente. Você pode basear-se no código-fonte Assembly **fibonacci.m1ps** e na planilha de acompanhamento **fibonacci.xls** na sua verificação.

Se seus projetos de Banco de Registradores (Parte **II**) e Memória (Parte **III**) estiverem corretos, você deverá ver os números de Fibonacci sendo exibidos no sinal *IO_OUT* a partir de, aproximadamente, metade do tempo de simulação.

ENTREGA: adicione no arquivo de entrega final um arquivo **fibonacci.png** com uma screenshot de simulação em que seja possível identificar a exibição de, pelo menos, dois números de Fibonacci consecutivos no sinal *IO_OUT*.

Parte V - Primeira execução do código de Fibonacci com mM

ATENÇÃO: Se estivermos ainda em aulas remotas, pule para a parte VI.

Defina novamente a entidade *mM_toplevel* como top-level do projeto. Compile e grave na placa.

Utilização do mM:

- Conecte a placa DE1-SoC a um monitor;
- SW0 serve para trocar a tela de visualização. Uma tela apresenta os valores dos registradores e os sinais internos da via de dados do processador. A outra apresenta o estado dos primeiros endereços da memória;
- SW[7..0] estão conectados aos bits menos significativos do sinal IO_IN do processador;
- Os visores de sete segmentos mostram os bits menos significativos do sinal IO_OUT do processador, convertidos para hexadecimal;
- KEY3 executa um *reset*;
- KEY2, quando pressionado, envia um clock de 20 Hz para o processador;
- KEY2, quando pressionado, envia um clock de 4 Hz para o processador;
- KEY0 é um clock manual.

Baseando-se no código Assembly **fibonacci.m1ps**, na planilha de acompanhamento **fibonacci.xls** e na simulação anterior (Parte **IV**), execute o código no processador (você terá que usar os switches da placa para enviar os valores de entrada) e confira o correto funcionamento.

DEMONSTRAÇÃO: A tarefa a ser demonstrada será sorteada, para cada grupo, dentre as 3 atividades de execução com o módulo mM (Partes **V**, **VII** e **IX**). Consulte o professor, PED ou PAD para saber qual tarefa foi sorteada para seu grupo.

Parte VI - Instrução ADDI

Modifique o processador para implementar a instrução ***ADDI***, conforme apresentado nos slides da Aula. Você está livre para fazer quaisquer alterações julgar necessárias no código do processador.

Formato da instrução ADDI:

Op(31..26)	Rd(25..21)	Rs1(20..16)	Imediato(15..0)	Obs
001 100				Rd ← Rs1 + Imed
6	5	5	16	

Dica: Concentre as modificações na via de dados (**CPU.bdf**) e unidade de controle (**UC.vhd**).

ENTREGA: Elabore um arquivo **addi.txt** descrevendo, resumidamente, a sua estratégia de projeto para a nova instrução: o que foi alterado no processador e por quê.

Parte VII - Execução do código de Fibonacci com ADDI

VII.1. Tomando como base o que foi feito na Parte **IV**, altere, na entidade *CPU* (**CPU.bdf)** o arquivo de imagem da memória de instruções para **fibonacci_addi.mif** e re-execute a simulação do processador. Utilize o código-fonte Assembly **fibonacci_addi.m1ps** e a planilha de acompanhamento **fibonacci_addi.xls** na sua verificação.

ENTREGA: Capture uma screenshot **fibonacci_addi.png** de simulação em que seja possível identificar a execução de, pelo menos, uma instrução ADDI.

VII.2. Tomando como base a Parte **V**, re-execute o programa de Fibonacci na placa utilizando a instrução ADDI.

DEMONSTRAÇÃO: A tarefa a ser demonstrada será sorteada, para cada grupo, dentre as 3 atividades de execução com o módulo mM (Partes **V**, **VII** e **IX**). Consulte o professor, PED ou PAD para saber qual tarefa foi sorteada para seu grupo.

Parte VIII - Instrução HALT

Implemente a instrução HALT no processador, que tem a seguinte especificação: ao executar esta instrução, a máquina de estados fica com o estado preso no ciclo de execução e somente via sinal externo de ***reset*** o processador reinicia a execução com PC=0.

Utilize as seguintes especificações:

- ***Opcode*** da instrução => 011000.
- Instrução => 0x60000000.

ENTREGA: Elabore um arquivo **halt.txt** descrevendo, brevemente, a sua estratégia de projeto para a nova instrução: o que foi alterado no processador e por quê.

Parte IX - Execução do código de Fibonacci com HALT

IX.1. Tomando como base o que foi feito na Parte **IV**, altere o arquivo de imagem da memória de instruções para **fibonacci_halt.mif** e re-execute a simulação do processador. Utilize o código-fonte Assembly **fibonacci_halt.m1ps** e a planilha de acompanhamento **fibonacci_halt.xls** na sua verificação.

ENTREGA: Capture uma screenshot **fibonacci_halt.png** de simulação em que seja possível identificar a execução da instrução HALT.

IX.2. Tomando como base a Parte **V**, re-execute o programa de Fibonacci na placa utilizando a instrução HALT.

DEMONSTRAÇÃO: A tarefa a ser demonstrada será sorteada, para cada grupo, dentre as 3 atividades de execução com o módulo mM (Partes **V**, **VII** e **IX**). Consulte o professor, PED ou PAD para saber qual tarefa foi sorteada para seu grupo.

- ENTREGA -

Entregue um único arquivo comprimido em formato **ZIP** de nome **F##Lab14.zip**, onde ## é o número da placa utilizada pelo grupo, contendo:

- Pasta **Processor** com todos os arquivos do processador originais e quaisquer outros criados para descrição do banco de registradores (parte **II**), da memória (parte **III**) e adição de ADDI (parte **VI**) e HALT (parte **VIII**).
- Arquivo **fibonacci.png** da parte **IV**.
- Arquivo **addi.txt** da parte **VI**.
- Arquivo **fibonacci_addi.png** da parte **VII**.
- Arquivo **halt.txt** da parte **VIII**.
- Arquivo **fibonacci_halt.png** da parte **IX**.