



UNIVERSIDADE FEDERAL DE SANTA CATARINA

Laboratório 1: Introdução ao Laboratório

EEL5105 – Circuitos e Técnicas Digitais

Objetivos

.....

- Introduzir o laboratório de **EEL5105**.
- Visão geral das **plataformas** e **ferramentas** utilizadas.
- Realizar projetos básicos de **circuitos digitais**.

EEL5105 – Circuitos e Técnicas Digitais

Introdução ao Laboratório

Tarefa

Tarefa Adicional

EEL5105 – Circuitos e Técnicas Digitais

.....

- **Professor:** Eduardo L. O. Batista

Os alunos podem entrar em contato a qualquer momento via e-mail para tirar dúvidas: <eduardo.batista@ufsc.br>.

Atendimento também via **Discord** (link no **Moodle**).

- **Monitoria:** Isabella Prando e Monique Hillesheins

<monitoria.eel5105@gmail.com>. Disponíveis para tirar dúvidas tanto de teoria quanto laboratório. Horário de atendimento estará disponível em breve no Moodle.

EEL5105 – Circuitos e Técnicas Digitais

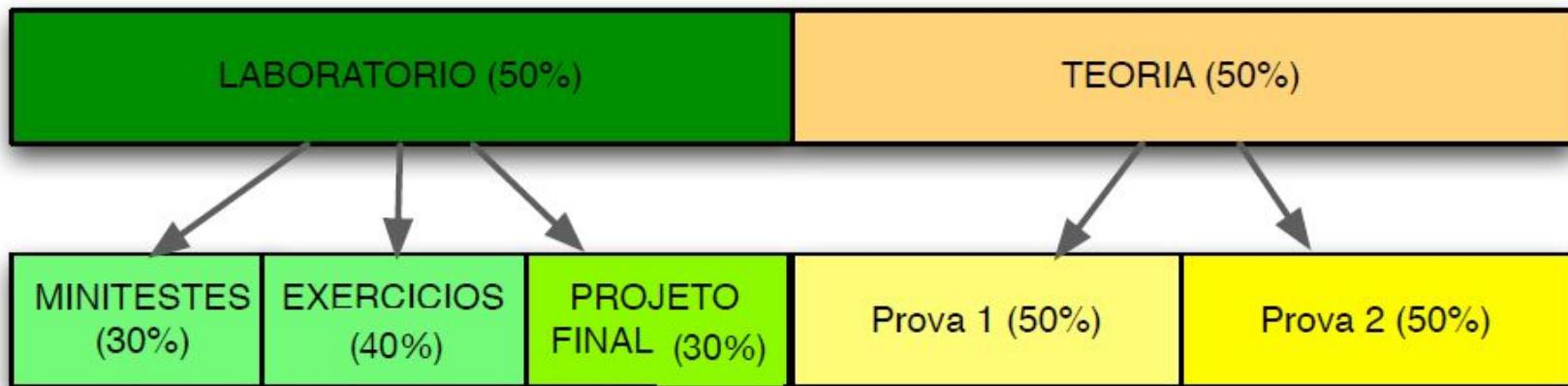
- Carga horária semanal
 - 2 horas-aula de **teoria**
 - 3 horas-aula de **laboratório**
- É oferecida para os seguintes cursos da **UFSC**:
 - Ciência da Computação
 - Engenharia Elétrica
 - Engenharia Eletrônica
 - Engenharia de Controle e Automação
 - Engenharia de Produção Elétrica

EEL5105 – Circuitos e Técnicas Digitais

- Nas aulas de laboratório: projeto, análise e síntese de circuitos/sistemas digitais descritos na linguagem **VHDL** e voltados para implementação em **FPGAs**.
- Três tipos de aulas:
 - **Aula de Laboratório:** Roteiros envolvendo tarefas a serem realizadas pelos alunos, além de **miniteste** a ser resolvido no **Moodle** com duração de 15 minutos.
 - **Aula de Exercício:** Um **problema específico** a ser resolvido pelo aluno no horário da aula com resultado enviado via **Moodle**.
 - **Projeto Final:** Projeto um pouco mais complexo a ser realizado ao longo das últimas semanas do semestre.

EEL5105 – Circuitos e Técnicas Digitais

- **Avaliação:**



EEL5105 – Circuitos e Técnicas Digitais

Introdução ao Laboratório

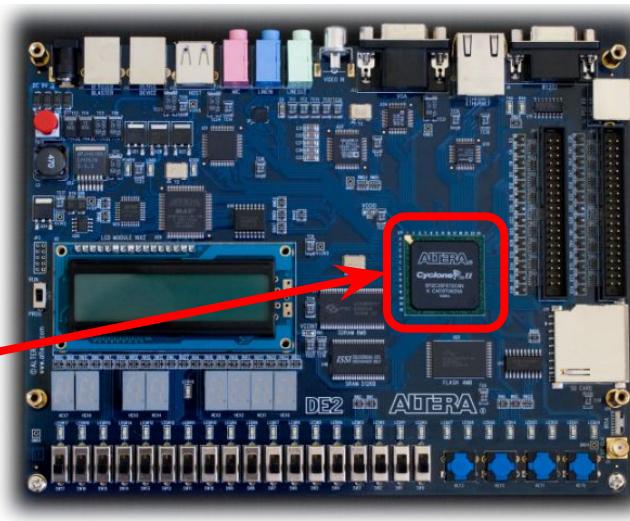
Tarefa

Tarefa Adicional

Introdução ao Laboratório

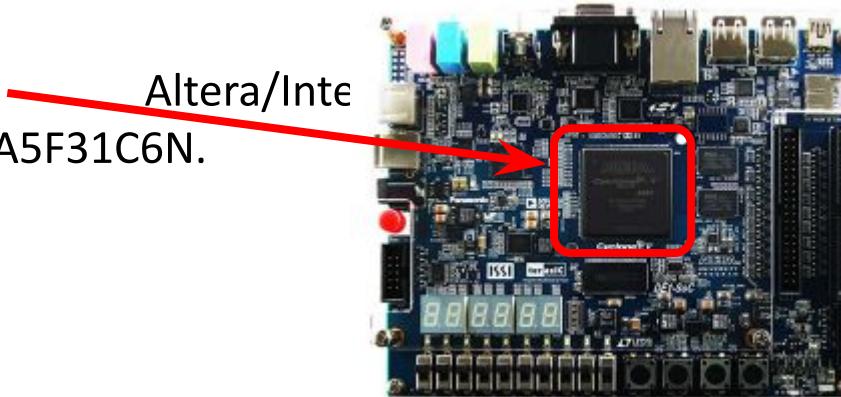
- Plataformas:

- Kit DE2
 - **FPGA** Altera/Intel
 - Cyclone II EP2C35F672C6



- Kit DE1-SoC

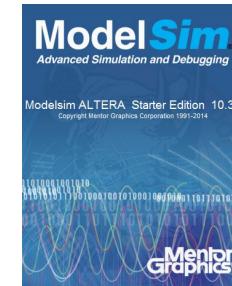
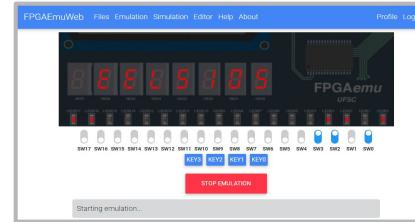
- **FPGA**
Altera/Inte
Cyclone V 5CSEMA5F31C6N.



Introdução ao Laboratório

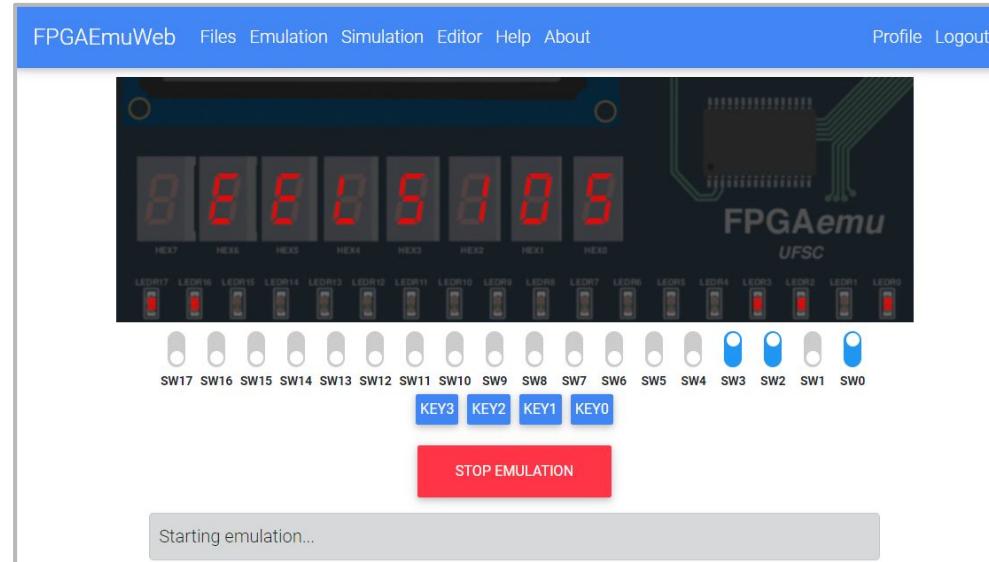
- **Ferramentas:**

- **Emuladores** desenvolvidos por professores da disciplina
(projeto/emulação/simulação)
- **Quartus II**
(projeto/síntese)
- **ModelSim**
(simulação)
- **EDAPlayground**
(simulação web)



Introdução ao Laboratório

- Ferramenta principal em nossas aulas:
 - **Emulador** de plataformas de desenvolvimento versão web (**FPGAEmuWeb**):

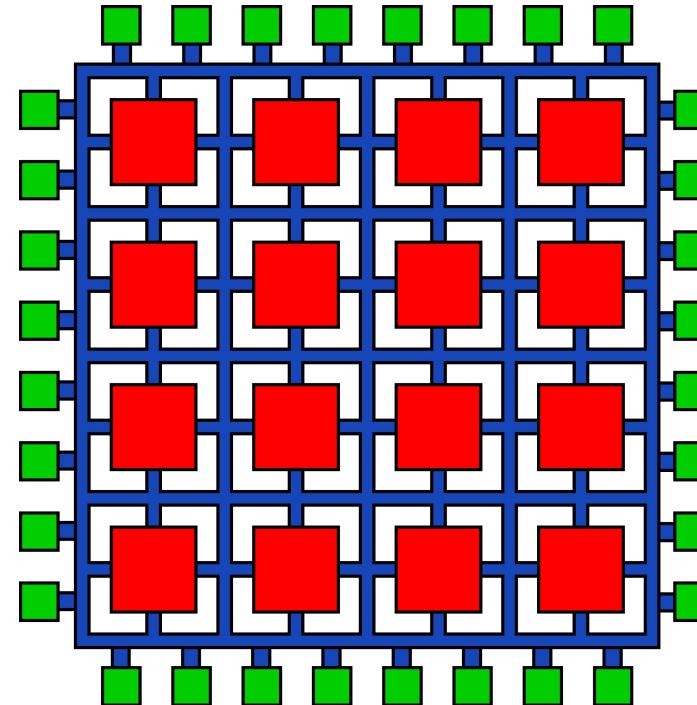


Introdução ao Laboratório

- **FPGA:** *field-programmable gate array*

- Estrutura interna
(muito simplificada):

 Blocos lógicos programáveis
 Entrada/saída
 Roteamento

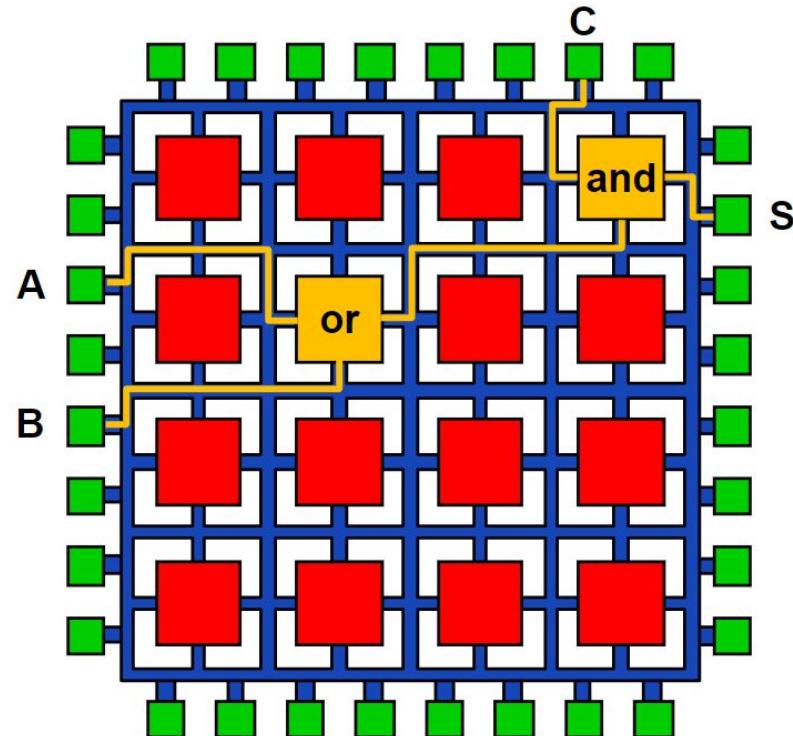


Introdução ao Laboratório

- **FPGA**: *field-programmable gate array*

- Pode ser “**programado**” em nível estrutural:

$$S = (A \text{ or } B) \text{ and } C$$



Introdução ao Laboratório

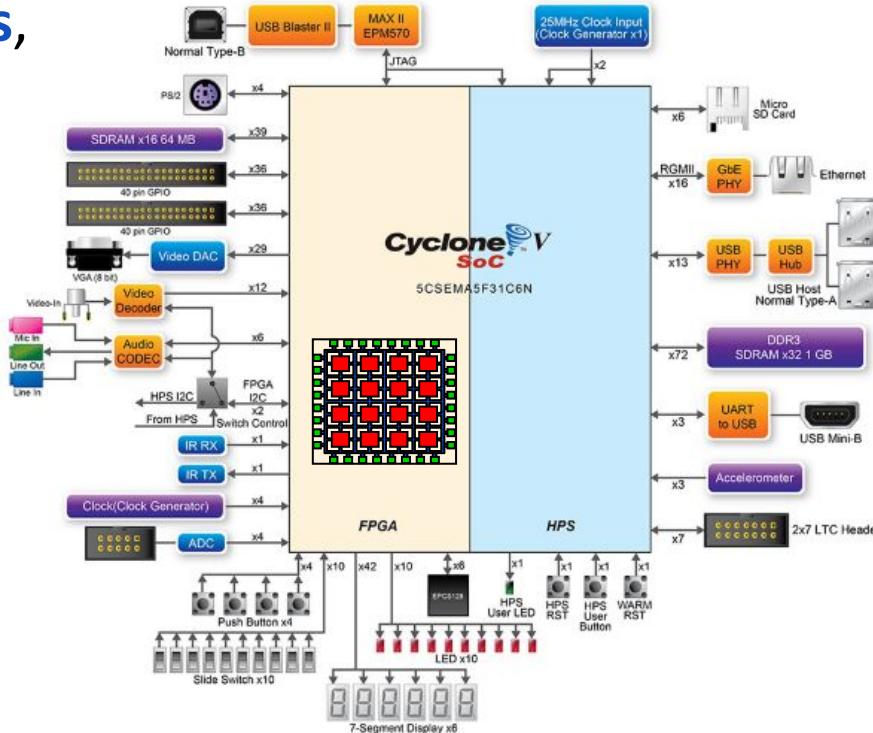
- **FPGA**: *field-programmable gate array*
 - Pode ser “**programado**” em nível estrutural.
- Enquanto um **Processador (CPU/GPU)** tem estrutura fixa e executa sequência de instruções, **FPGA** pode ser reconfigurado completamente. Assim,
 - **CPU/GPU**: execução em geral sequencial, mais fácil de programar, desempenho mais limitado.
 - **FPGA**: alto grau de paralelismo (uma vez que o código em geral não é executado sequencialmente, mas gera arquitetura no chip), maior dificuldade de projetar, desempenho tende a ser melhor.
- Video interessante: <https://www.youtube.com/watch?v=8POZhFHxBLs>

Introdução ao Laboratório

- Cada um tem seu papel:
 - **CPU/GPU**: execução em geral sequencial, mais fácil de programar, desempenho mais limitado.
 - Partes de mais alto nível de um sistema.
 - **FPGA**: alto grau de paralelismo (uma vez que o código em geral não é executado sequencialmente, mas gera arquitetura no chip), maior dificuldade de projetar, desempenho tende a ser melhor.
 - Partes críticas em desempenho.
- Maiores detalhes:
<https://blog.esciencecenter.nl/why-use-an-fpga-instead-of-a-cpu-or-gpu-b234cd4f309c>

Introdução ao Laboratório

- Nos kits **DE2** e **DE1-SoC**, os pinos do **FPGA** estão ligados em **chaves, botões, leds, displays de 7 segmentos, etc**, permitindo que estímulos sejam enviados ao **FPGA** e saídas sejam observadas de diferentes formas.



Introdução ao Laboratório

- Exemplos no kit DE2:

- Displays de 7-seg



- Botões

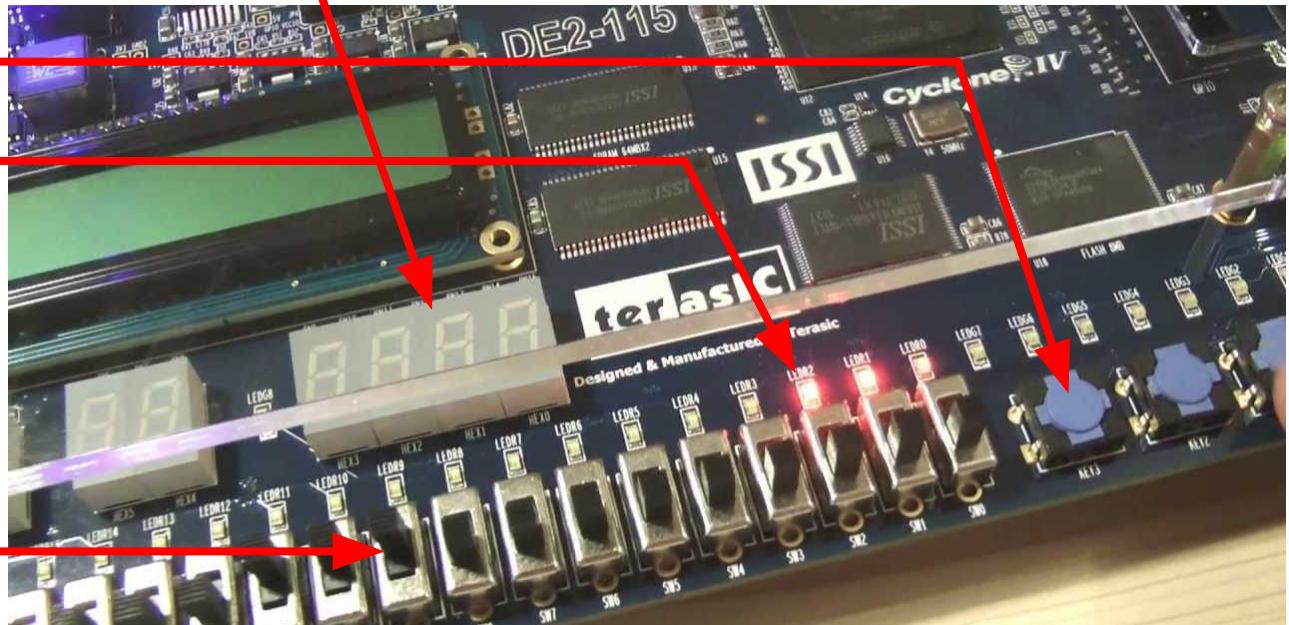


- Leds



- Chaves

- liga/desliga



Introdução ao Laboratório

- Exemplos no **Emulador Web:**

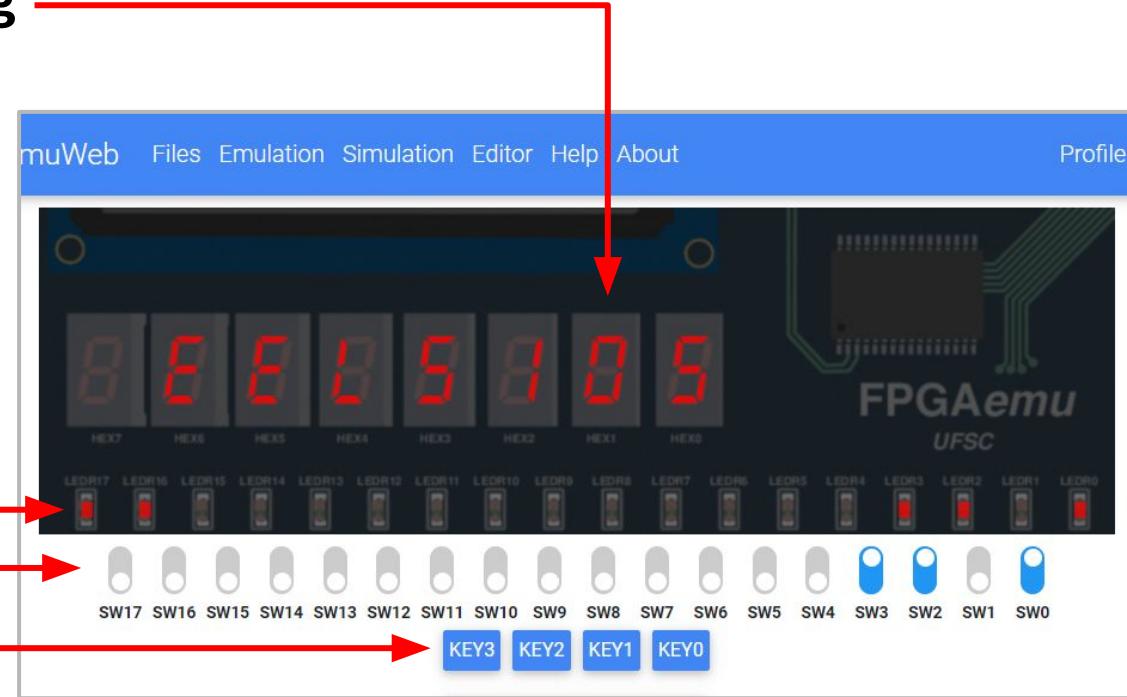
- Displays de 7-seg

- Botões

- Leds

- Chaves

- liga/desliga



Introdução ao Laboratório

- Linguagem **VHDL**
 - *Very-High-Speed Integrated-Circuit Hardware Description Language*
 - Uma das principais **linguagens de descrição de hardware**.
(Atenção: não é linguagem de programação!)
 - É a linguagem que usaremos para fazer a **síntese** de circuitos em FPGAs ou nos emuladores.
 - **Atenção:** iremos descrever arquiteturas de hardware, o que é **diferente de execução sequencial de código**. Na verdade, cada **linha ou parte do código** irá descrever uma **parte de um circuito**.

EEL5105 – Circuitos e Técnicas Digitais

Introdução

Tarefas: Familiarização com o Emulador

Tarefa Adicional

Tarefas

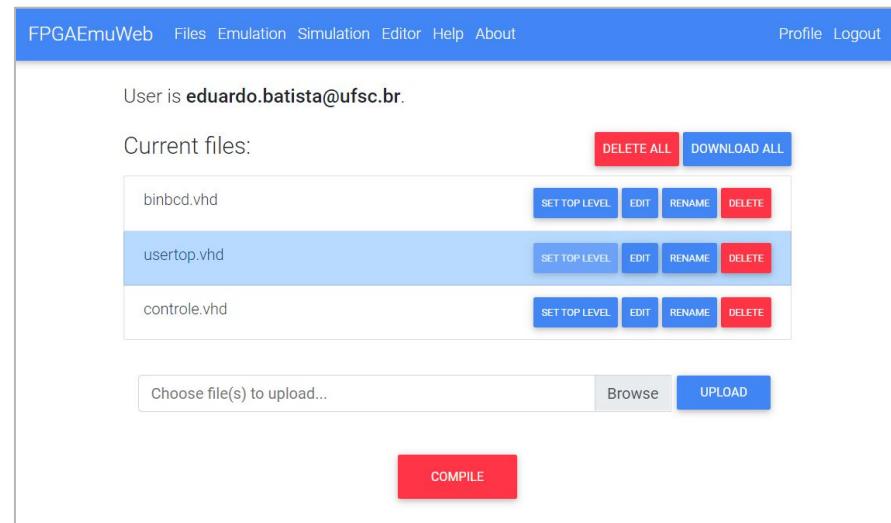
.....

- O foco nesta primeira aula é promover a familiarização do aluno com o **Emulador** e também permitir um primeiro contato com a linguagem **VHDL**.
- Para acessar o emulador, acesse o link disponível no **Moodle** da disciplina e clique em **Sign Up** para se cadastrar:



Tarefas

- Itens do **menu principal** do emulador:
 - **Files**: gerência de arquivos;
 - **Emulation**: emulação;
 - **Simulation**: simulação;
 - **Editor**: editor de código;
 - **Help**: ajuda rápida;
 - **About**: sobre;
 - **Profile**: perfil do usuário;
 - **Logout**: sair do sistema.



Tarefas

- **Tarefa 1:** Conectando **chaves**, **keys** e **leds**.
 - Para essa primeira tarefa, inicialmente acesse o item **Editor** do **FPGAEmuWeb**, clique no botão **New** para criar um novo arquivo e dê o nome “**usertop.vhd**” para tal arquivo.
 - A seguir, clique no botão **Templates** e selecione **usertop.vhd**. Com isso, o arquivo que você criou será preenchido com um código base em **VHDL** onde estão definidos, como entradas e saídas do sistema (**ports**), os elementos **LEDR** (com 18 bits), **SW** (também com 18 bits), **KEY** (com 4 bits) e **HEX0** até **HEX7** (cada um com 7 bits).

Tarefas

- **Tarefa 1:** Conectando **chaves**, **keys** e **leds**.
 - A definição dessas **ports** resulta no seguinte mapeamento pelo emulador:
 - **LEDR** para os 18 leds vermelhos;
 - **SW** para as 18 chaves liga/desliga;
 - **KEY** para os 4 botões (*push buttons*);
 - e **HEX0** a **HEX7** para os 8 displays de 7 segmentos.
 - Clique então no botão **Save** para salvar o seu código e em seguida no botão **Compile** para que a **síntese** do projeto seja realizada.



Tarefas

- **Tarefa 1:** Conectando **chaves, keys e leds.**
 - Terminada a compilação, clique em **Emulation** no menu principal e depois no botão **Start Emulation**.
 - A mensagem “**EEL5105**” deverá então aparecer nos displays de 7 segmentos indicando que o seu **usertop.vhd** está sendo emulado.
 - Com emulação rodando, acione as chaves **SW0** a **SW3** no emulador e observe o que acontece com os leds vermelhos **LEDR0** a **LEDR3** do sistema. Acione também os botões **KEY0** e **KEY1** e observe o que acontece com **LEDR16** e **LEDR17**.
Atenção: ao acionar **KEY0** e **KEY1**, mantenha o botão do mouse pressionado.

Tarefas

- **Tarefa 1:** Conectando **chaves, keys e leds.**
 - Agora, pare a emulação (*Stop Emulation*) e volte para o editor de código para ver o **usertop.vhd**. Analisando esse código, em particular a parte ilustrada no próximo slide, é possível observar as ligações entre chaves/keys/leds/displays que resultam no comportamento observado na emulação.

Tarefas

- Tarefa 1: Conectando **chaves**, **keys** e **leds**.

```
architecture rtl of usertop is
begin

    LEDR(0) <= SW(0);
    LEDR(1) <= SW(1);
    LEDR(2) <= SW(2);
    LEDR(3) <= SW(3); }

    LEDR(16) <= KEY(0);
    LEDR(17) <= KEY(1);
    :
    HEX3 <= "0010010";
    HEX2 <= "1111001";
    HEX1 <= "1000000";
    HEX0 <= "0010010"; }

end rtl;
```

Ligações de leds com chaves. Poderia ser simplesmente:
LEDR(3 downto 0) <= SW(3 downto 0);

Ligações de leds com botões. Poderia ser simplesmente:
LEDR(17 downto 16) <= KEY(1 downto 0);

Escreve **5105** no display de 7 segmentos.
Mais detalhes na **Tarefa 2**.

Tarefas

- **Tarefa 2:** Acionando os Displays de 7 Segmentos
 - A ideia agora em entender como funciona o acionamento dos displays de 7 segmentos. Observando o trecho do código da Tarefa 1 correspondente ao acionamento dos displays de 7 segmentos (destacado abaixo), nota-se que palavras de 7 bits são usadas para controlar cada display, sendo que cada bit nessa correspondência ao display, não de cada segmento. Veja slide para entender melhor.

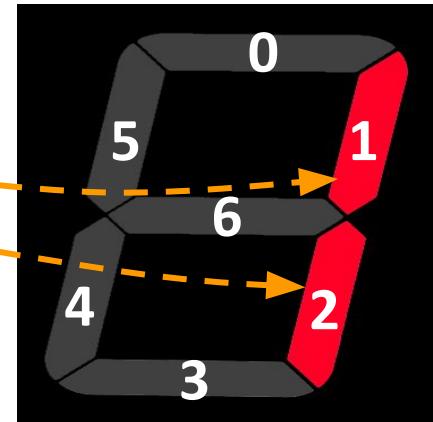
```
HEX6 <= "0000110";  
HEX5 <= "0000110";  
HEX4 <= "1000111";  
HEX3 <= "0010010";  
HEX2 <= "1111001";  
HEX1 <= "1000000";  
HEX0 <= "0010010";
```

Tarefas

- **Tarefa 2:** Acionando os Displays de 7 Segmentos
 - A figura abaixo ilustra o acionamento de cada segmento do display com definições em palavra ou bit a bit para o display de número 2 (**HEX2**).
 - **Atenção:** note que um bit definido como **0** resulta no **acendimento** do segmento (segmentos são ativos-baixos, acendem para nível lógico baixo ou **0**).

```
-- Mostra número 1:  
HEX2 <= "1111001";
```

```
-- Mostra número 1,  
-- outra opção:  
HEX2(0) <= '1';  
HEX2(1) <= '0';  
HEX2(2) <= '0';  
HEX2(3) <= '1';  
HEX2(4) <= '1';  
HEX2(5) <= '1';  
HEX2(6) <= '1';
```



Tarefas

- **Tarefa 2:** Acionando os Displays de 7 Segmentos
 - Nesta tarefa você deve modificar o código da **Tarefa 1** visando modificar o acionamento dos displays **HEX3** a **HEX0** para que a palavra **UFSC** apareça em tais displays.
 - **Atenção:** em VHDL, as linhas devem ser terminadas com ponto-e-virgula (;).
 - **Atenção 2:** note, nos códigos de bits individuais são feitas com aspas simples ('0' ou '1') enquanto que aspas duplas são utilizadas para múltiplos bits (palavras): "0100011".



Tarefas

- **Tarefa 3:** Conversor Binário para BCD
 - O objetivo agora é observar o funcionamento de um circuito que converte uma entrada **binária** de 8 bits no **código BCD** correspondente (com 12 bits).
 - Antes da emulação do circuito, você deve ser capaz de responder as seguintes perguntas:
 - 1) Por que, para uma entrada binária de 8 bits, a saída BCD tem 12 bits?**
 - 2) Qual seria a representação binária para o decimal 34? E para o decimal 255?**
 - 3) Qual seria a representação BCD para o decimal 34? E para o decimal 255?**

Tarefas

- **Tarefa 3:** Conversor Binário para BCD
 - Baixe o arquivo **binbcd.vhd** disponível no **Moodle** e faça o upload para o **FPGAEmuWeb** a partir da aba **Files**.
 - **Atenção:** não se preocupe em entender o código de **binbcd.vhd**, apenas avalie o funcionamento em termos de entradas e saídas.
 - Defina **binbcd.vhd** como arquivo principal para emulação, clicando no botão **Set Top Level** correspondente.
 - Faça a “**compilação**” e **emulação** observando que:
 - A entrada binária está em **SW7** a **SW0** (8 bits).
 - A saída BCD está em **LEDR11** a **LEDRO** (12 bits).

Código

retirado

de:

<https://stackoverflow.com/questions/23871792/convert-8bit-binary-number-to-bcd-in-vhdl>

Tarefas

- Tarefa 3: Conversor Binário para BCD



Tarefas

- **Sugestão**

- Terminado o roteiro, fique à vontade para testar novas conexões entre os elementos de entrada e saída do emulador modificando o código usado nas Tarefas 1 e 2.