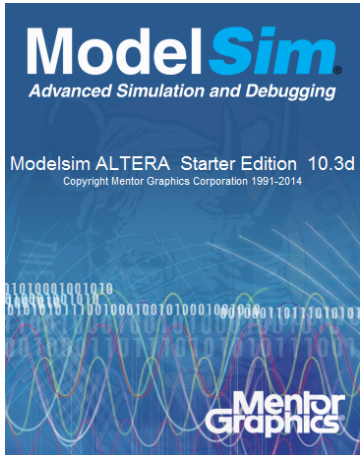
FEDERAL UNIVERSITY
OF SANTA CATARINA

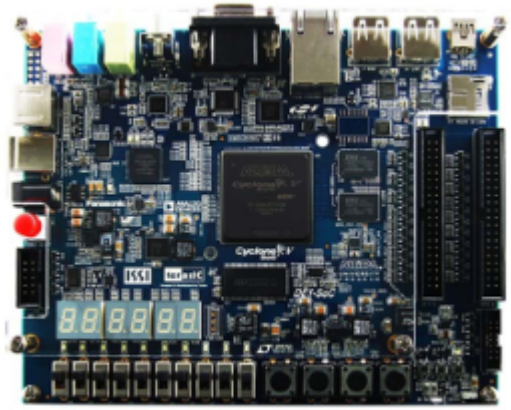
# Laboratório 2:
# Introdução à VHDL e Modelsim

**EEL5105 – Circuitos e Técnicas Digitais**

# Objetivos

- Apresentar uma visão geral sobre **VHDL** e estudar **exemplos de descrição de hardware** em VHDL
- Familiarização com o simulador utilizado na disciplina ***ModelSim***
- Compreensão do fluxo de projeto de sistemas digitais: Projeto, **simulação** e prototipação

# Introdução

Tarefa

Tarefa Adicional

# Introdução

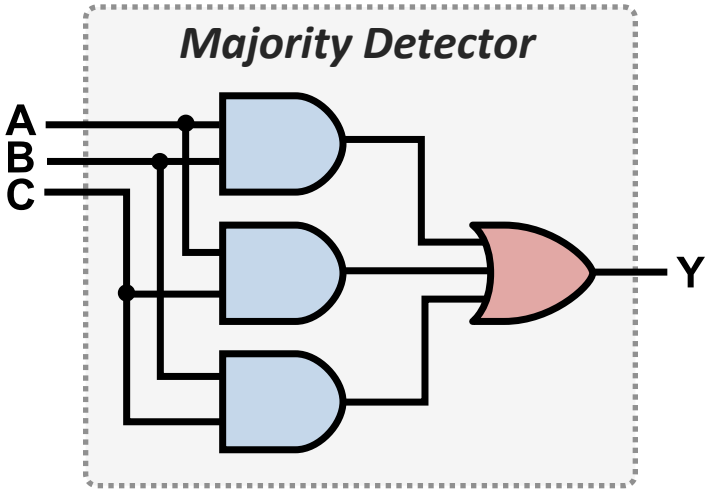
- **VHDL - Visão Geral**
  - **VHDL** é uma linguagem para descrição de hardware
  - **VHDL** = **V**HSIC **H**ardware **D**escription **L**anguage
  - No final da década de 80, **VHDL** se tornou uma linguagem padrão para o **IEEE** (***Institute of Electrical and Electronic Engineers***).
  - Existem diversas ferramentas para simular e sintetizar (gerar hardware) circuitos descritos em **VHDL**.
  - Outras linguagens de descrição de hardware: Verilog, SystemC, AHDL, Handel-C, System Verilog, Abel, Ruby, ...

# Introdução

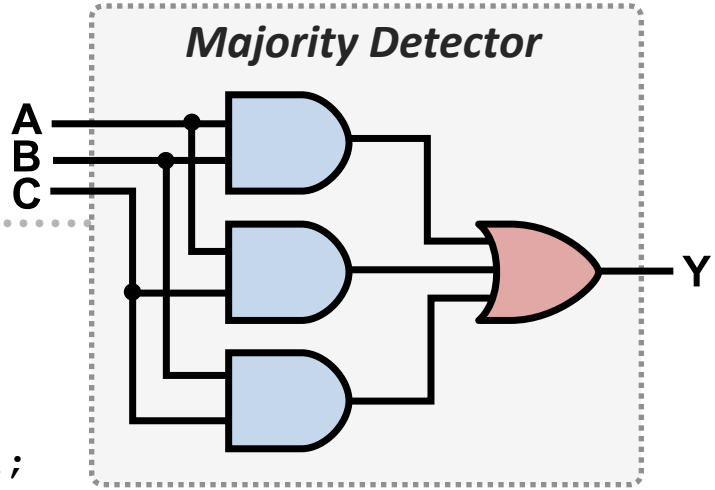
- **VHDL - Visão Geral**
  - O projeto de um circuito digital pode ser descrito em **VHDL** em diversos níveis de abstração (ex.: **estrutural**, **comportamental**).
  - Descrições em VHDL podem ser utilizadas para gerar **hardware** (arquivo para configuração de um FPGA, por exemplo).
  - Descrições em VHDL podem ser **simuladas** (executadas em um simulador).
  - A geração de estímulos para simulação VHDL é realizada por intermédio de **testbenches** (como feito na aula anterior). Um **testbench** define os estímulos externos a serem utilizados como entrada para o circuito.

# Introdução

- **VHDL – Exemplo de código: *Majority Detector***



# Introdução




- VHDL – *Majority Detector*

**LIBRARIES**

```
library IEEE;
use IEEE.Std_Logic_1164.all;
```

**ENTITY**

```
entity majority is
port (A: in std_logic;
      B: in std_logic;
      C: in std_logic;
      Y: out std_logic
      );
end majority;
```

**ARCHITECTURE**

```
architecture circuito_logico of majority is
  signal D,E,F: std_logic;
begin
  Y <= D or E or F;
  D <= A and B;
  E <= A and C;
  F <= B and C;
end circuito_logico;
```

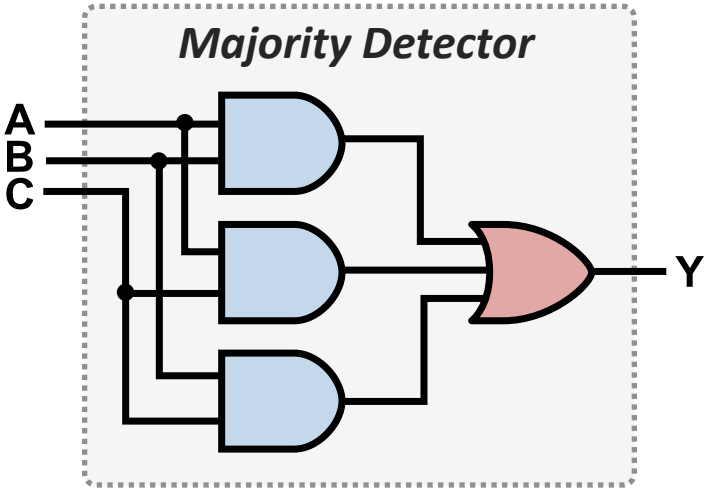
# Introdução

- **VHDL – *Majority Detector***
  - **LIBRARIES** : bibliotecas necessárias

```
library IEEE;
use IEEE.Std_Logic_1164.all;
```

# Introdução

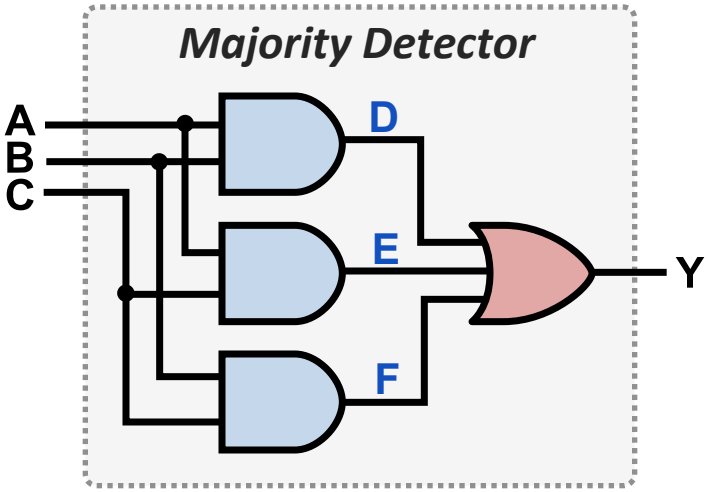
- VHDL – *Majority Detector*
  - **ENTITY** : define os **pinos** do circuito digital, ou seja, a **interface** entre a lógica implementada e o mundo externo.




```
entity majority is
port (A: in std_logic;
      B: in std_logic;
      C: in std_logic;
      Y: out std_logic
      );
end majority;
```

# Introdução

- **VHDL – *Majority Detector***
  - **ARCHITECTURE** : define a funcionalidade do circuito digital, utilizando os **pinos** de entrada e saída listados na **ENTITY**, além de **signals** para fazer as conexões internas.




```
architecture circuito of majority is
  signal D,E,F: std_logic;
begin
  Y <= D or E or F;
  D <= A and B;
  E <= A and C;
  F <= B and C;
end circuito;
```

Introdução

# Tarefa

Tarefa Adicional

# Tarefa

- Projetar, **simular** e implementar um ***Majority Detector***
  - ***Majority Detector***: saída em nível lógico alto sempre que a maioria dos bits de entrada estiver em nível lógico alto

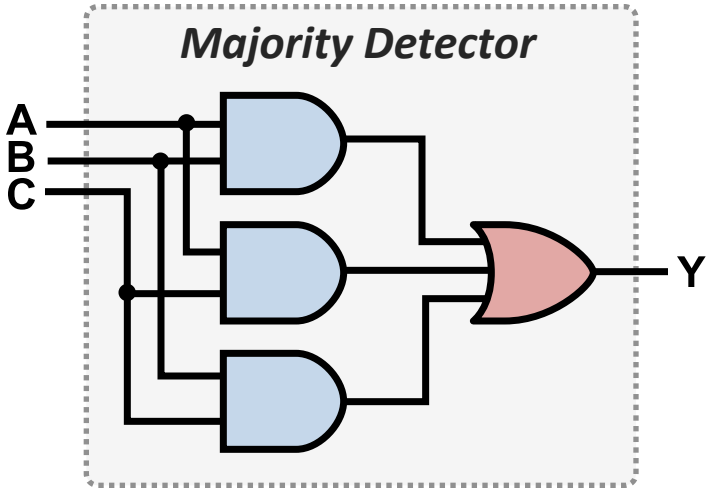



Tabela verdade

| A B C | Y |
|---|---|
| 0 0 0 | 0 |
| 0 0 1 | 0 |
| 0 1 0 | 0 |
| 0 1 1 | 1 |
| 1 0 0 | 0 |
| 1 0 1 | 1 |
| 1 1 0 | 1 |
| 1 1 1 | 1 |

**Y = (A and B) or (A and C) or (B and C)**

# Tarefa

- ***Majority Detector***

  - **Passo 1**: Criar projeto no **Quartus II**
    - Acessar **File -> New Project Wizard** e criar um projeto como feito na aula anterior.
      - Sugestão de nome do projeto: **Lab2**
      - Dispositivo: **Cyclone V**, **5CSEMA5F31C6**

  - **Passo 2**: Criar arquivo do tipo **VHDL** dentro do projeto.
    - **File -> New -> Design Files -> VHDL File**
    - **File -> Save As** com algum nome desejado (ex.: "***Majority.vhd***")
    - Com arquivo aberto: **Project -> Set As Top Level Entity**

**Atenção**: nome do arquivo deve ser igual ao da **ENTITY** que você vai criar.

# Tarefa

- ***Majority Detector***
  - **Passo 3**: Escreva o código **VHDL** do ***Majority*** no arquivo criado.
    - **Libraries**:

```
library IEEE;
use IEEE.Std_Logic_1164.all;
```

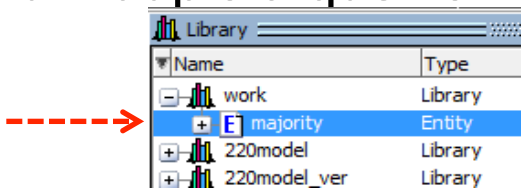
    - **Entity**:

```
entity majority is
port (A: in std_logic;
      B: in std_logic;
      C: in std_logic;
      Y: out std_logic
      );
end majority;
```

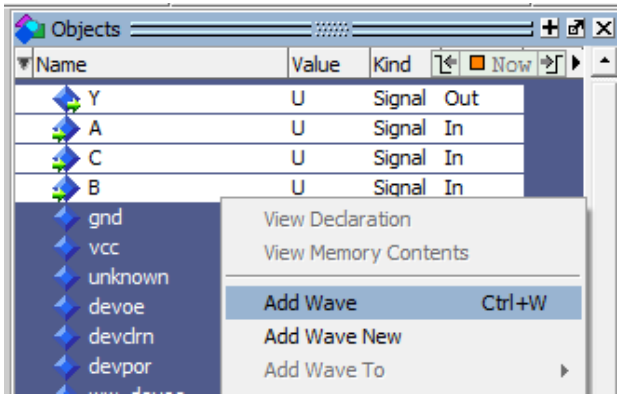
    - Escrever a **Architecture (ver slide 10)**

# Tarefa

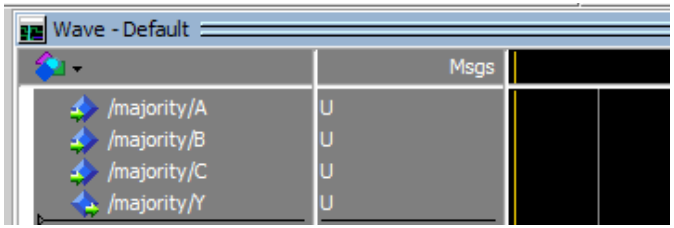
- **Passo 4**: Simulação no ***ModelSim***
  - Use **Tools -> Run Simulation Tool -> RTL Simulation** no **Quartus II**
  - Dê um duplo clique no módulo a ser simulado no grupo **work**:




  - Selecione os sinais de interesse (**A**, **B**, **C** e **Y**) na janela **Objects**, e adicione eles à simulação usando a opção **Add Wave**:

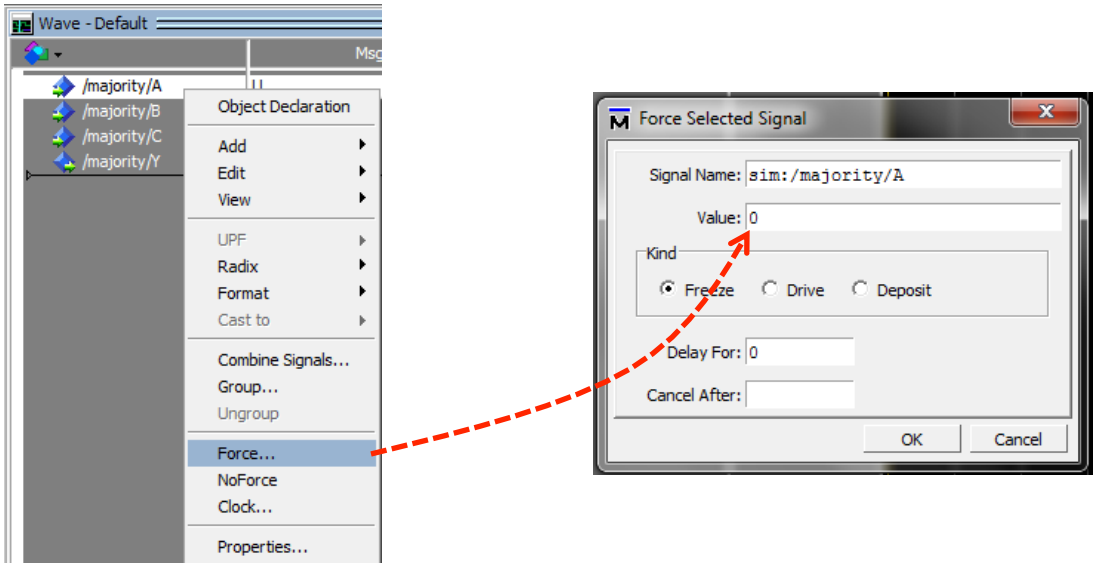



  - Com isso, os sinais de interesse deverão aparecer na janela **Wave**:



# Tarefa

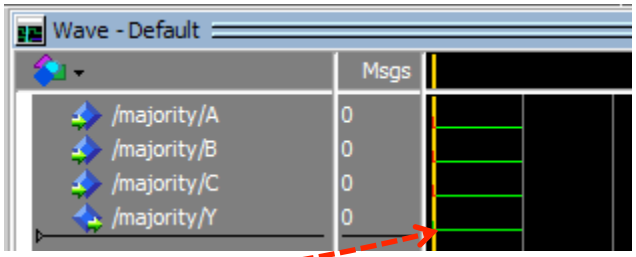
- **Passo 4**: Simulação no ***ModelSim***
  - Utilize a opção ***Force*** para atribuir valores às entradas **A**, **B** e **C** do circuito:




  - Atribua inicialmente os valores **A = B = C = 0** e, em seguida, clique em **Simulate -> *Run*** (ou simplesmente pressione **F9**) para simular o circuito com tais valores nas entradas.

# Tarefa

- **Passo 4**: Simulação no ***ModelSim***
  - Como resultado, você deverá observar uma linha verde em nível lógico baixo para a saída **Y**, uma vez que, para o circuito considerado, **Y = 0** para **A = B = C = 0**:




  - Simule então agora o circuito para todas as combinações possíveis de valores das entradas e complete a **tabela verdade**:

| A B C | Y |
|---|---|
| 0 0 0 | 0 |
| 0 0 1 | |
| 0 1 0 | |
| 0 1 1 | |
| 1 0 0 | |
| ↓ | ↓ |

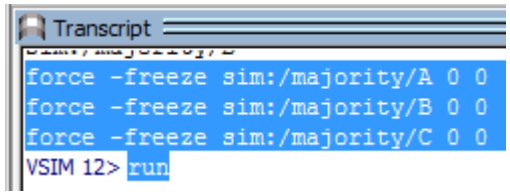
## Tarefa

- **Passo 4**: Simulação no ***ModelSim***

  - Finalmente, compare a **tabela verdade obtida** com a **tabela verdade esperada para o seu circuito** e verifique se ele está funcionando corretamente.

# Tarefa

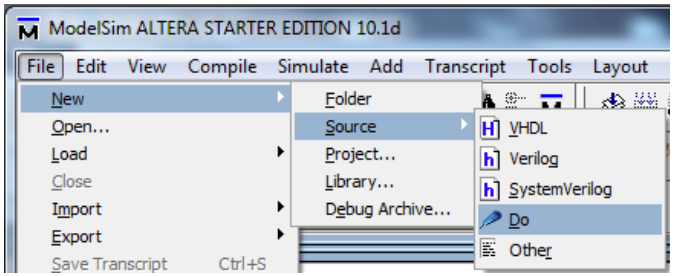
- **Passo 5**: Automatizando a Simulação
  - O procedimento de simulação pode ser automatizado com a criação de um **script de simulação (testbech)**.
  - Para tal, force **A = B = C = 0**, rode sua simulação (pressionando **F9**) e, na janela **Transcript**, você poderá observar a sequencia de quatro comandos utilizada para forçar valores e rodar a simulação:



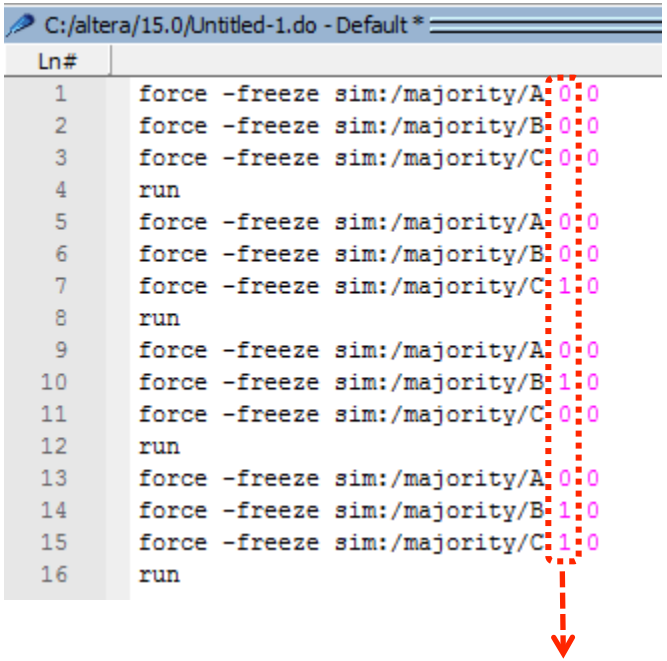

  - Para criar um **script** a partir de tais comandos, inicialmente você deve selecionar tais comandos e copiá-los (**Ctrl+C**).

# Tarefa

- **Passo 5**: Automatizando a Simulação
  - Cria agora um arquivo de **script** usando **File -> New -> Source -> Do**

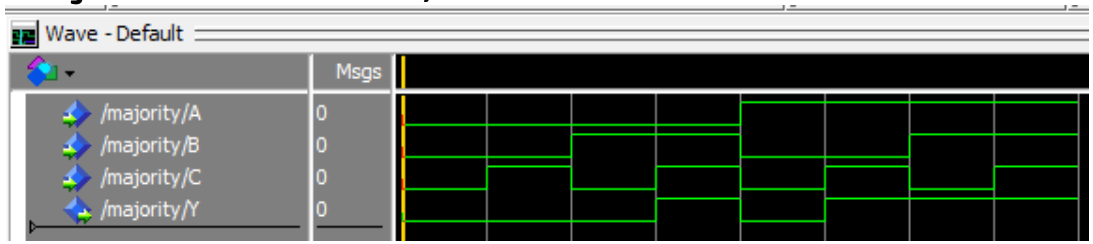



  - Cole os comandos copiados e modifique-os para que a simulação seja executada com diferentes combinações de valores para **A**, **B** e **C**:

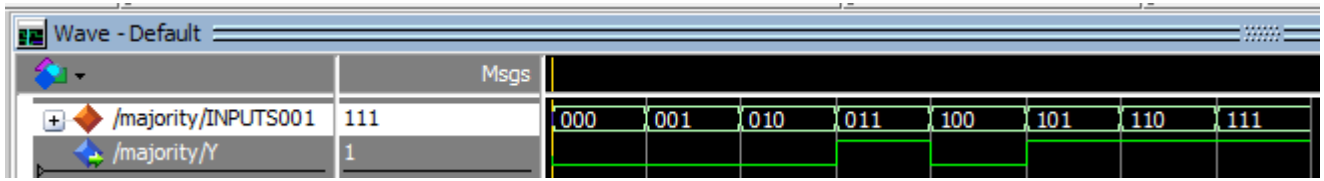


# Tarefa

- **Passo 5**: Automatizando a Simulação
  - Salve seu script (por exemplo, como **sim1.do**) e, para executá-lo, digite o seguinte comando na janela **Transcript**: **do sim1.do**
  - Como resultado, sua simulação deverá rodar para todas as combinações de valores, resultando em

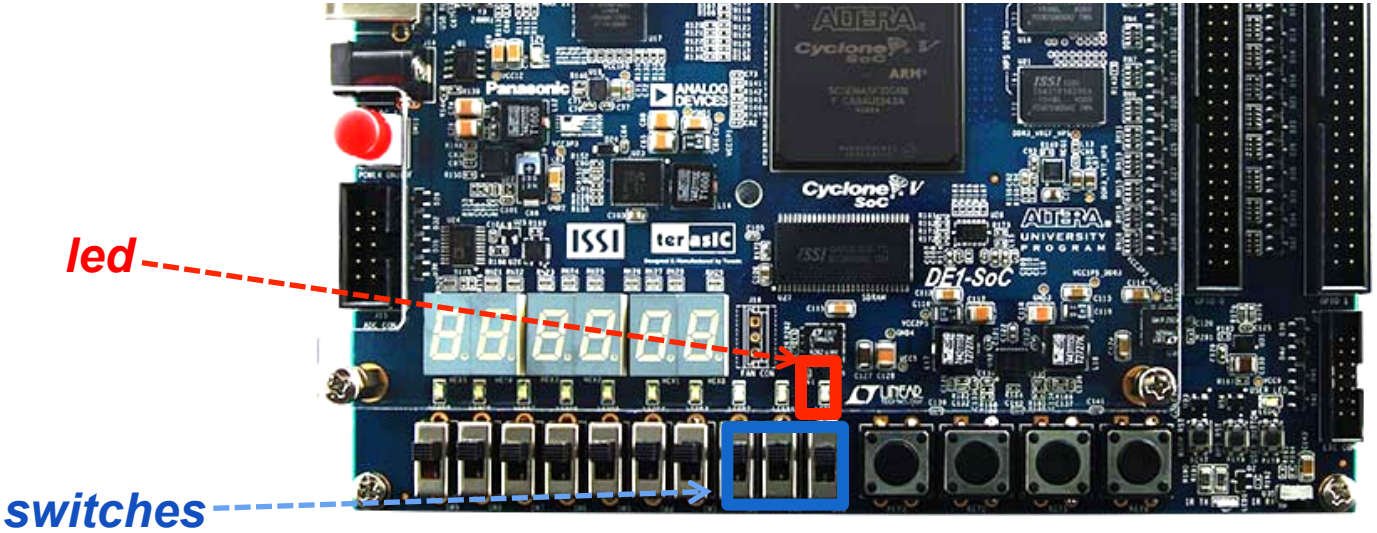



  - Para facilitar a visualização dos resultados, você pode selecionar os três sinais de entrada (**A**, **B** e **C**) e usar a opção **Combine Signals** para criar um agrupamento de nome **INPUTS**, resultando em:



# Tarefa

- **Passo 6**: Fazer a prototipação em **FPGA** do ***Majority Detector***
  - Com isso, o fluxo de **projeto, simulação e prototipação** é completado
- Para tal, associe **A**, **B**, **C**, e **Y** a **SW(0)**, **SW(1)**, **SW(2)** e **LEDR(0)**, como visto na aula anterior, faça novamente a síntese e programe o FPGA.



# Tarefa

- Mapeamento:

| Pino | Elemento |
|---|---|
| A | SW[0] |
| B | SW[1] |
| C | SW[2] |
| Y | LEDR[0] |

- Seguir os passos **5** e **6** da aula anterior para associação de **entradas/saídas** com **chaves/leds** usando arquivo **Pinos.qsf** disponível no Moodle e prototipação no **DE1-SOC**.

# Tarefa Avançada

# Tarefa Avançada

- Modifique o circuito anterior como apresentado a seguir. Dito circuito detecta um valor entre 0 e 7. Determine o valor de entrada que o circuito detecta a partir do analise da simulação.

| A | B | C | Valor |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 2 |
| 0 | 1 | 1 | 3 |
| 1 | 0 | 0 | 4 |
| 1 | 0 | 1 | 5 |
| 1 | 1 | 0 | 6 |
| 1 | 1 | 1 | 7 |