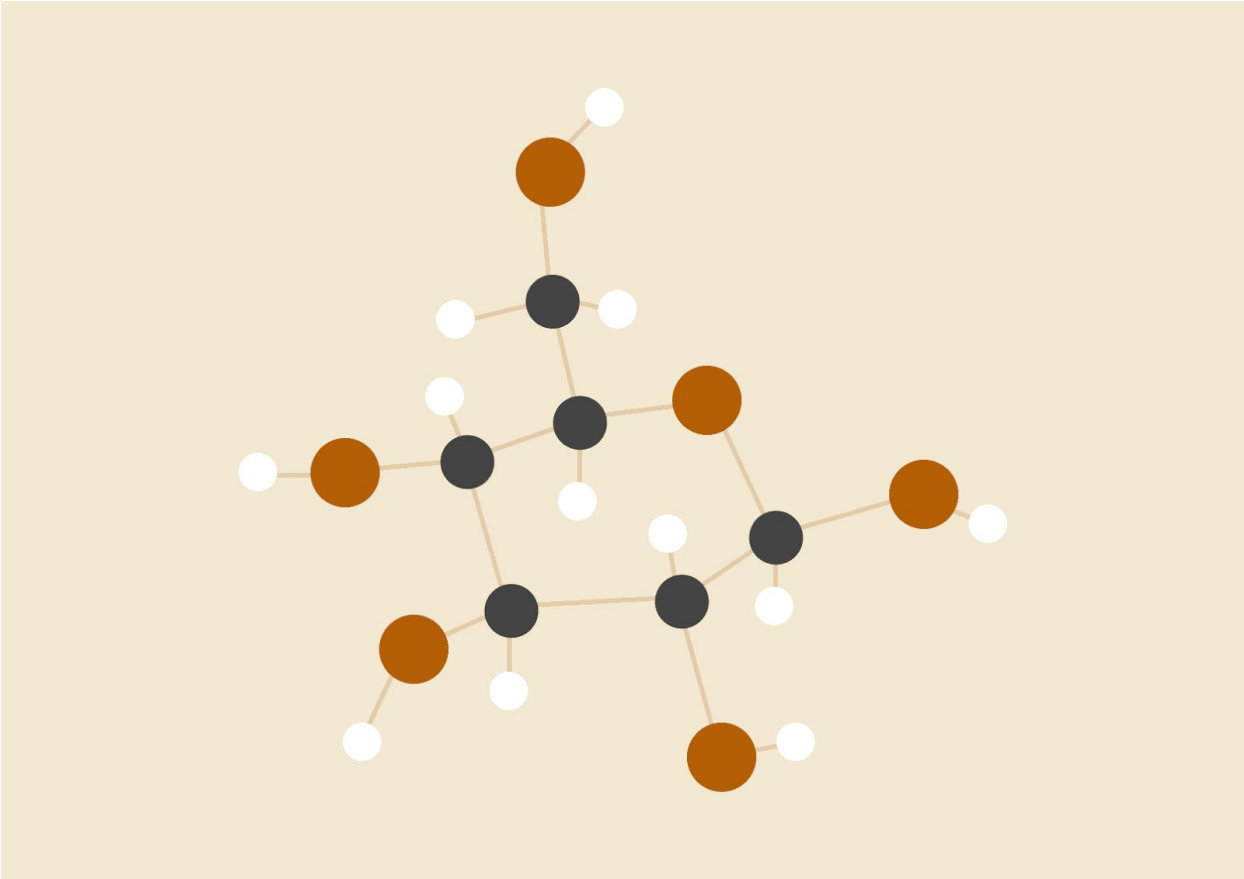
RELATÓRIO
SISTEMAS DIGITAIS

Desenvolvimento de uma Calculadora em VHDL



Fabrício Ribeiro Bueno Yamamoto - 11036814

12.08.2018

Prof. Rodrigo Bacurau

INTRODUÇÃO

A calculadora desenvolvida tem os seguintes funcionalidades:

- Capaz de realizar soma e subtração.
- Capacidade de utilização do último resultado como operando da próxima operação.
- Capacidade de realizar operações com operandos de até 12 bits.
- Capacidade de operar com números inteiros positivos e negativos no formato de complemento de 2.
- Indicação de overflow e underflow.
- Apresentação do resultado nos 4 displays de 7 segmentos.
- Controle de brilho dos displays de 7 segmentos.
- Entrada de dados pelo teclado com interface PS2.

MANUAL DE UTILIZAÇÃO

Trata-se de uma calculadora que funciona da seguinte maneira:

Entre com um operando de até 12 bits (-2048 até 2047), então entre com um operador (+ ou -) e pressione “enter” para ver o resultado.

Ao entrar com um operador logo após uma operação é possível utilizar o valor retornado pela operação previamente realizada.

Caso o “enter” seja pressionado após uma operação ele a repetirá com último valor recebido.

É possível controlar o brilho do display com as chaves da placa (SW0 até SW7).

Caso o número seja negativo, luzes LEDG0 ao LEDG7 irão acender.

Caso a operação realizada resulte em *overflow* ou *underflow*, luzes LEDR0 ao LEDR9 irão acender, e será necessário resetar a placa.

Aperte KEY1 para resetar a placa.

ORGANIZAÇÃO DO PROJETO

O projeto foi dividido em 8 arquivos .vhd diferentes:

- **calculadora**.vhd
  - **Descrição**: implementa a lógica da calculadora e importa todos os componentes, além de executar toda parte de IO.
  - **Entrada**: todo mapeamento de input do pin planner.
  - **Saída**: todo mapeamento de output pin planner.
- **binary_to_bcd**.vhd
  - **Descrição**: conversão de número binário para binary-coded decimal (BCD), utilizando-se o algoritmo double dabble.
  - **Entrada**: um número binário de 12 bits (STD_LOGIC_VECTOR).
  - **Saída**: quatro números binários de 4 bits (STD_LOGIC_VECTOR).
- **conv_7seg**.vhd
  - **Descrição**: conversão de um número binário para representação no display de 7 segmentos.
  - **Entrada**: um número binário de 4 bits (STD_LOGIC_VECTOR).
  - **Saída**: um vetor de 7 bits (STD_LOGIC_VECTOR) com as informações para o HEX[0~3] da placa.
- **conv_calc**.vhd
  - **Descrição**: conversão da entrada do teclado PS2 para número/comando/operador.
  - **Entrada**: um vetor de 8 bits (STD_LOGIC_VECTOR), que representa parte do scancode do teclado PS2.
  - **Saída**: um vetor de 4 bits (STD_LOGIC_VECTOR), relacionado ao número/comando/operador digitado.

- **PWM**.vhd
    - **Descrição**: controlador da intensidade do brilho do display de 7 segmentos.
    - **Entrada**:
        - bit (STD_LOGIC), que representa o clock.
        - bit (STD_LOGIC), que representa o reset.
        - bit (STD_LOGIC), que representa o enable.
        - vetor de 8 bits (STD_LOGIC_VECTOR), que representa o DUTY, relacionado diretamente ao SW[0~7] da placa.
        - vetor de 16 bits (STD_LOGIC_VECTOR), que representa os 4 dígitos de 4 bits que irão aparecer no display de 7 segmentos.
    - **Saída**: um vetor de 16 bits (STD_LOGIC_VECTOR), que representa os 4 dígitos de 4 bits que irão aparecer no display de 7 segmentos, porém com o brilho controlado de acordo com o DUTY.

Para utilização do teclado foi importado uma biblioteca compostas pelos seguintes arquivos:

- **kbdex_ctrl**.vhd
- **keyboard**.vhd
- **ps2_iobase**.vhd

Algoritmos implementados, porém não integrados:

- **testaSomador**.vhd
    - **Descrição**: componente que recebe duas entradas, retorna sua soma e se houve overflow/underflow.
    - **Entrada**: dois números binários de **n** bits (STD_LOGIC_VECTOR) e um GENERIC **n** (tamanho do vetor de bits).
    - **Saída**: um número binário de **n** bits (STD_LOGIC_VECTOR) e um bit (STD_LOGIC).
    - **Observação**: este componente importa os seguintes outros:
        - **SomadorBinarioParalelo**.vhd
        - **SomadorBinarioCompleto**.vhd

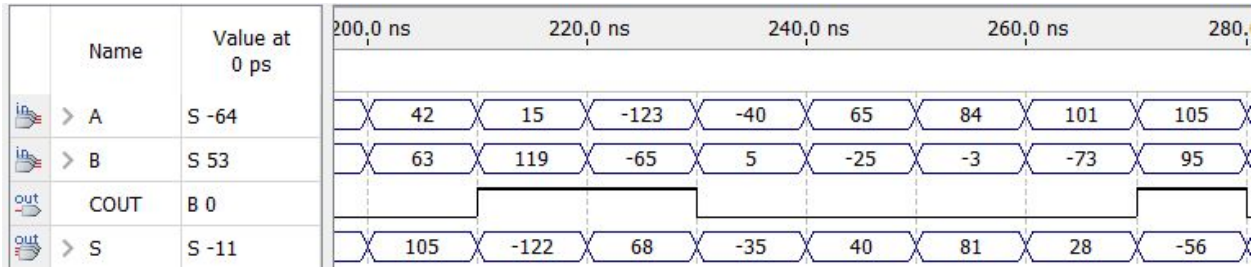
Pela falta de integração, hoje o projeto executa sua soma de forma interna com a biblioteca STD_LOGIC_UNSIGNED.

PIN PLANNER

To	Direction	Location
CLOCK_24	Input	PIN_A12
CLOCK_24	Input	PIN_B12
CLOCK_27	Input	PIN_E12
CLOCK_27	Input	PIN_D12
CLOCK_50	Input	PIN_L1
HEX0[6]	Output	PIN_E2
HEX0[5]	Output	PIN_F1
HEX0[4]	Output	PIN_F2
HEX0[3]	Output	PIN_H1
HEX0[2]	Output	PIN_H2
HEX0[1]	Output	PIN_J1
HEX0[0]	Output	PIN_J2
HEX1[6]	Output	PIN_D1
HEX1[5]	Output	PIN_D2
HEX1[4]	Output	PIN_G3
HEX1[3]	Output	PIN_H4
HEX1[2]	Output	PIN_H5
HEX1[1]	Output	PIN_H6
HEX1[0]	Output	PIN_E1
HEX2[6]	Output	PIN_D3
HEX2[5]	Output	PIN_E4
HEX2[4]	Output	PIN_E3
HEX2[3]	Output	PIN_C1
HEX2[2]	Output	PIN_C2
HEX2[1]	Output	PIN_G6
HEX2[0]	Output	PIN_G5
HEX3[6]	Output	PIN_D4
HEX3[5]	Output	PIN_F3
HEX3[4]	Output	PIN_L8
HEX3[3]	Output	PIN_J4
HEX3[2]	Output	PIN_D6
HEX3[1]	Output	PIN_D5
HEX3[0]	Output	PIN_F4
KEY[3]	Input	PIN_T21
KEY[2]	Input	PIN_T22
KEY[1]	Input	PIN_R21
KEY[0]	Input	PIN_R22
LEDG[7]	Output	PIN_Y21
LEDG[6]	Output	PIN_Y22
LEDG[5]	Output	PIN_W21
LEDG[4]	Output	PIN_W22
LEDG[3]	Output	PIN_V21
LEDG[2]	Output	PIN_V22
LEDG[1]	Output	PIN_U21
LEDG[0]	Output	PIN_U22
LEDR[9]	Output	PIN_R17
LEDR[8]	Output	PIN_R18
LEDR[7]	Output	PIN_U18
LEDR[6]	Output	PIN_Y18
LEDR[5]	Output	PIN_V19
LEDR[4]	Output	PIN_T18
LEDR[3]	Output	PIN_Y19
LEDR[2]	Output	PIN_U19
LEDR[1]	Output	PIN_R19
LEDR[0]	Output	PIN_R20
PS2_CLK	Bidir	PIN_H15
PS2_DAT	Bidir	PIN_J14
SW[9]	Input	PIN_L2
SW[8]	Input	PIN_M1
SW[7]	Input	PIN_M2
SW[6]	Input	PIN_U11
SW[5]	Input	PIN_U12
SW[4]	Input	PIN_W12
SW[3]	Input	PIN_V12
SW[2]	Input	PIN_M22
SW[1]	Input	PIN_L21
SW[0]	Input	PIN_L22

PROBLEMAS IDENTIFICADOS E NÃO RESOLVIDOS

A soma atualmente é feita via biblioteca STD_LOGIC_UNSIGNED, pois não obtive sucesso ao integrar o meu algoritmo de SomadorBinarioParalelo. Este que está funcionando perfeitamente de forma isolada, com implementação de GENERIC, inclusive indicando *overflow* e *underflow*, comprovado por testes.



Testes do componente TestaSomador.vhd (n = 8 bits), presente no projeto, mas não integrado.

Era possível fazer multiplicação e subtração com a utilização de SIGNED, entretanto, ao tentar aplicar o multiplicador com algoritmo de Booth, essa funcionalidade foi retirada da calculadora. E, infelizmente, até o momento (20 de agosto 2018) não consegui finalizar a implementação do algoritmo de Booth.

Aparentemente o sistema do distribuidor do software Quartus II 13.0sp1 está fora do ar até hoje (20 de agosto 2018) para recuperação de senha, me impossibilitando de fazer o download do software na minha casa em São Paulo, tendo apenas em meu notebook, que fica em Santo André, o que me atrapalhou consideravelmente durante um final de semana inteiro, que codifiquei apenas utilizando um editor de texto.

AUTORIA DOS CÓDIGOS UTILIZADOS

A princípio, a ideia foi reutilizar os códigos feitos no decorrer da disciplina.

- **calculadora**.vhd
  - Código original.
- **binary_to_bcd**.vhd
  - Referências de: https://en.wikipedia.org/wiki/Double_dabble.
  - Utilizei como referência e apliquei modificação autoral para números binários do tipo SIGNED serem mostrados da devida forma.
- **PWM**.vhd
  - Código original.
- **conv_calc**
  - Código original.
- **conv_7seg**.vhd
  - Utilizado na biblioteca do teclado PS2 disponível no site da disciplina, com uma pequena modificação para integrar com o PWM.
- **kbdex_ctrl**.vhd
  - Utilizado na biblioteca do teclado PS2 disponível no site da disciplina.
- **keyboard**.vhd
  - Utilizado na biblioteca do teclado PS2 disponível no site da disciplina.
- **ps2_iobase**.vhd
  - Utilizado na biblioteca do teclado PS2 disponível no site da disciplina.

Algoritmos implementados, porém não integrados:

- **testaSomador**.vhd - Código original
  - **SomadorBinarioParalelo**.vhd - Código original
  - **SomadorBinarioCompleto**.vhd - Código original

CÓDIGOS DESENVOLVIDOS

O código foi feito em cima do esqueleto do projeto **PS2_KEYBOARD** por praticidade na hora de reutilizar o pin planner e a própria biblioteca.

O projeto como um todo pode ser encontrado no meu GitHub:

- https://github.com/fabriciorby/VHDL